W3School ios教程

wizardforcel

Published with GitBook



目錄

W3School ios教程

来源：ios教程

整理：飞龙

IOS 简介

IOS之前被称为iPhone OS，是一个由苹果公司开发的移动操作系统。

iOS的第一个版本是在2007年发布的，其中包括iPhone和iPod Touch。

2004年4月发布iPad（第一代），并于2012年11月发布了iPad迷你款。

IOS设备发布相当频繁，由以往经验可知，每年都会推出至少一个版本的iPhone和iPad。

现在发布了iPhone5s，之前还推出了iPhone，iPhone3gs，iPhone4,iPhone4s以及iphone5。

同样的iPad也从iPad一代更新到iPad四代以及一个特别的迷你版iPad。

iOS SDK已经从1.0更新到6.0。最新的iOS SDK6.0，是唯一支持Xcode4.5和其更高版本的版本。

丰富的苹果文档，使我们能找到许多方法和库用于我们的部署目标。在Xcode的当前版本中，我们能够在iOS4.3,5.0和6.0的部署目标之间选择。

IOS的影响能够从以下的特点显现:

Facebook和Twitter上，加速度计，GPS，高端处理器，相机，Safari浏览器，功能强大的API，游戏中心，在应用程序内购买，提醒，宽范围的手势

- 地图
- Siri
- Facebook 和 Twitter
- Multi-Touch（多点触摸）
- Accelerometer（加速度传感器）
- GPS
- 高性能处理器
- 相机
- Safari浏览器
- 功能强大的API
- 游戏中心
- 在应用程序内购买
- 提醒功能
- 手势

iPhone和iPad的用户日益增多，这为iPhone和iPad应用商城的研发者创造了赚钱的机遇。

IOS最新的一点是，苹果公司研发了应用商城，这样用户可以购买应用程序来完善他们的iOS设备。

研发者可以在应用商城发布免费和付费的应用软件。

开发应用程序并将其发布到应用商店，开发人员需要注册iOS开发者计划，为其发展更新Xcode每年话费99美元和Mac Mountain Lion 或更高。

注册Apple开发者

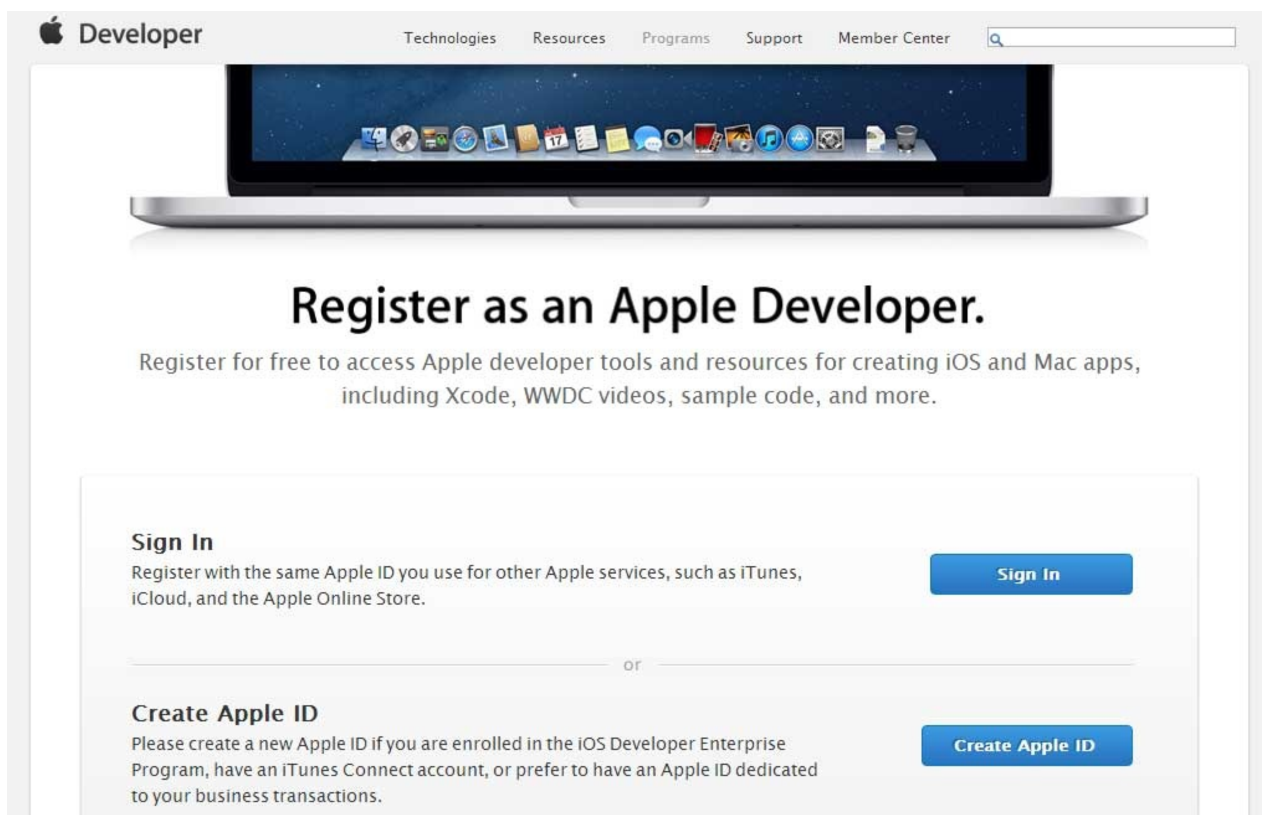
对拥有Apple设备的用户来说，非常有必要拥有Apple ID，而且成为一个研发者，必须用到Apple ID,获取 Apple ID是免费的，也无需有资费方面的顾虑。

拥有Apple账户有以下好处：

- 易于了解研发工具；
- 全球研发者视频会议；
- 受邀加入iOS研发者团队;

注册苹果账号

1、单击 (https://developer.apple.com/programs/register/) 并选择创建Apple ID



2、输入个人信息

3、返回邮箱确认，激活账号

4、下载研发工具，Xcode及它所包含的iOS模拟器，iOS SDK和其他研发资源

申请APP开发者

1、点击 (https://developer.apple.com/programs/ios/)



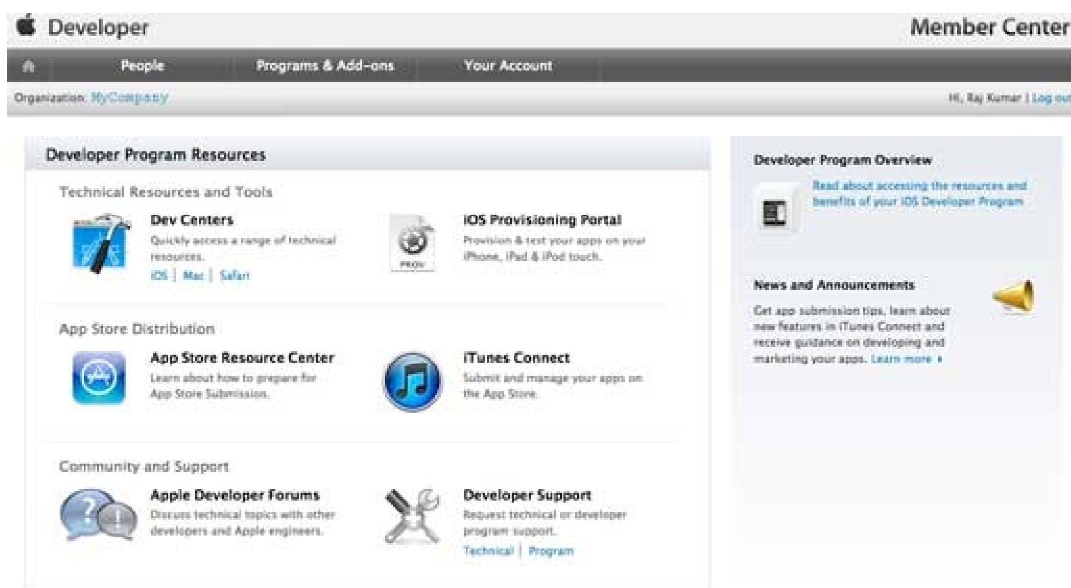
2、点击注册页面

3、登录账号（已有账号）或注册Apple ID

4、选择个人账号或公司账号，研发者团队使用公司账号，个人账号不能添加其他用户

5、新用户进入个人信息页面，使用信用卡购买加入研发项目

6、选择会员中心，利用研发者资源



7、在此处可以执行以下操作:

- 创建资源调配的配置文件
- 管理团队和设备
- 通过iTunes Connect管理应用到应用程序
- 获取论坛和技术支持

IOS Xcode 安装

1、从 https://developer.apple.com/downloads/ 下载Xcode的最新版本。



2、双击Xcode dmg文件

3、将找到的设备安装和打开

4、在这里会有两个项目在显示的窗口中即Xcode应用程序和应用程序文件夹的快捷方式

5、将Xcode拖拽并复制到应用程序

6、在应用里选择和运行程序，Xcode也将成为运行程序中的一部分

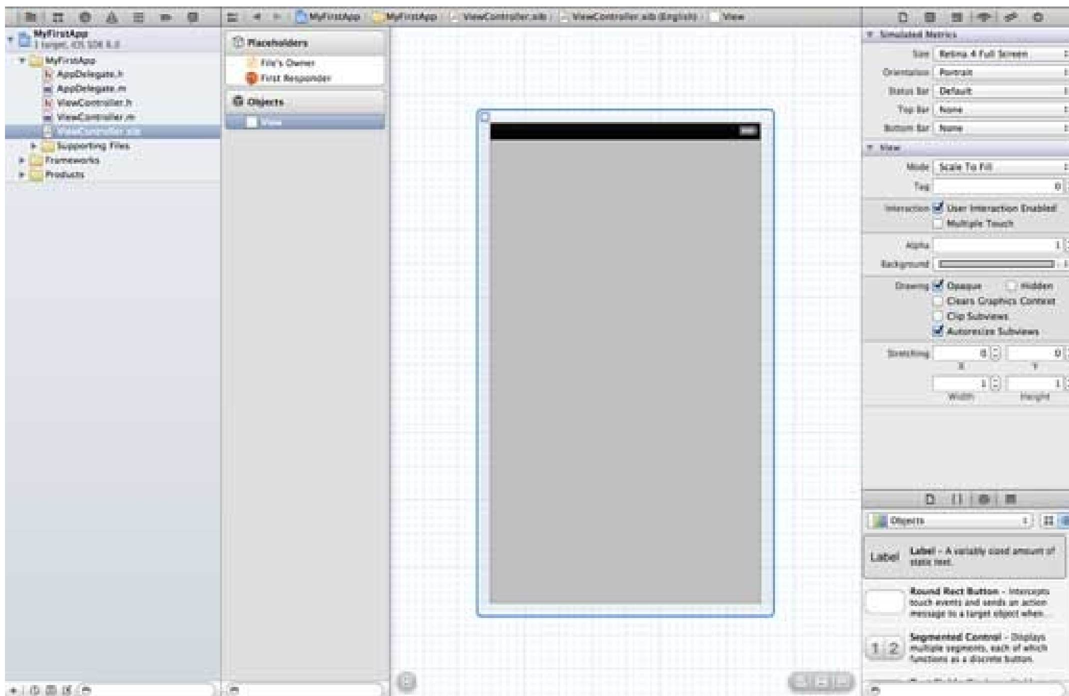
还可以从Mac App store里下载Xcode，并按照屏幕上的安装步

界面生成器（Interface Builder）

利用界面生成器这一工具，能很容易的创建UI界面。

可利用一系列的UI元素，拖拽进入UI可视界面。

我们将在接下来的页面了解添加用户界面元素，创建零售商和UI元素的操作。



在对象库的下方包含有全部必要的UI元素。用户界面通常称为xibs，这是他们的文件扩展名。

每个xibs都链接到相应的试图控制器。

IOS模拟器

IOS模拟器实际上包含两种类型的设备即iPhone和iPad及其不同的版本。

iPhone版本包括iPhone（常规版）、iPhone Retina，iPhone5,iPhone53。

Ipad有iPad和iPad Retina。iPhone模拟器显示如下:



你可以在经度和纬度影响应用程序的位置的情况下运行iOS模拟器，也可以模拟内存警告和呼叫在模拟器中的状态。

能够多数目的使用模拟器，但不能测试像加速度计这样的设备的功能。因此你可能需要iOS设备来测试一个应用程序的所有方面。

Objective-C 简介

在iOS的开发中使用的是Objective C语言，它是一中面向对象的语言，因而对于已经掌握面向对象语言知识的编程者来说是非常简单的。

接口和实现

在Objective里完成的文件被称为界面文件，该类文件的定义被称为实现文件。

一个简单的界面文件MyClass.h将如图所示:

```
@interace MyClass:NSObject{
// class variable declared here
}
// class properties declared here
// class methods and instance methods declared here
@end
```

执行MyClass.m文件，如下所示

```
@implementation MyClass
// class methods defined here
@end
```

创建对象

完成创建对象，如下所示

```
MyClass  *objectName = [[MyClass alloc]init] ;
```

方法（methods）

Objective C中声明的方法如下所示:

```
-(returnType)methodName:(typeName) variable1 :(typeName)variable2;
```

下面显示了一个示例:

```
-(void)calculateAreaForRectangleWithLength:(CGfloat)length
andBreadth:(CGfloat)breadth;
```

你可能会想什么是andBreadth字符串，其实它的可选字符串可以帮助我们阅读和理解方法，尤其是当方法被调用的时候。

在同一类中调用此方法，我们使用下面的语句。

```
[self calculateAreaForRectangleWithLength:30 andBreadth:20];
```

正如上文所说的andBreath使用有助于我们理解breath是20。Self用来指定它是一个类的方法。

类方法（**class methods**）

直接而无需创建的对象，可以访问类方法。他们没有任何变量和它关联的对象。示例如下:

```
+(void)simpleClassMethod;
```

它可以通过使用类名（假设作为MyClass类名称）访问，如下所示:

```
[MyClass simpleClassMethod];
```

实例方法

可以创建的类的对象后只访问实例方法，内存分配到的实例变量。实例方法如下所示:

```
-(void)simpleInstanceMethod;
```

创建类的对象后，它可以访问它。如下所示:

```
MyClass  *objectName = [[MyClass alloc]init] ;
[objectName simpleInstanceMethod];
```

Objective C的重要数据类型

数据类型
NSString 字符串
CGfloat 浮点值的基本类型
NSInteger 整型
BOOL 布尔型

打印日志

NSLog用于打印一份声明，它将打印在设备日志和调试版本的控制台和分别调试模式上。

如 NSlog(@"");

控制结构

除了几个增补的条款外，大多数的控制结构与C以及C++相同

属性（**properties**）

用于访问类的外部类的变量属性

比如：@property（非原子、强）NSString*myString

访问属性

可以使用点运算符访问属性，若要访问上一属性可以执行以下操作

```
self.myString = @"Test";
```

还可以使用set的方法，如下所示：

```
[self setMyString:@"Test"];
```

类别（**categories**）

类用于将方法添加到现有类。通过这种方法可以将方法添加到类，甚至不用执行文件，就可以在其中定义实际的类。MyClass的样本类别，如下所示：

```
@interace MyClass(customAdditions)
- (void)sampleCategoryMethod;
@end

@implementation MyClass(categoryAdditions)

-(void)sampleCategoryMethod{
   NSLog(@"Just a test category");
}
```

数组（arrays）

NSMutable和NSArray 是ObjectiveC中使用的数组类，前者是可变数组，后者是不可变数组。如下:

```
NSMutableArray *aMutableArray = [[NSMutableArray alloc]init];
[anArray addObject:@"firstobject"];
NSArray *aImmutableArray = [[NSArray alloc]
initWithObjects:@"firstObject",nil];
```

词典

NSMutableDictionary和NSDictionary是Objective中使用的字典，前者可变词典，后者不可变词典，如下所示:

```
NSMutableDictionary*aMutableDictionary = [[NSMutableArray alloc]ini
[aMutableDictionary setObject:@"firstobject" forKey:@"aKey"];
NSDictionary*aImmutableDictionary= [[NSDictionary alloc]initWithObj
@"firstObject",nil] forKeys:[ NSArray arrayWithObjects:@"aKey"]];
```

创建第一款iPhone应用程序

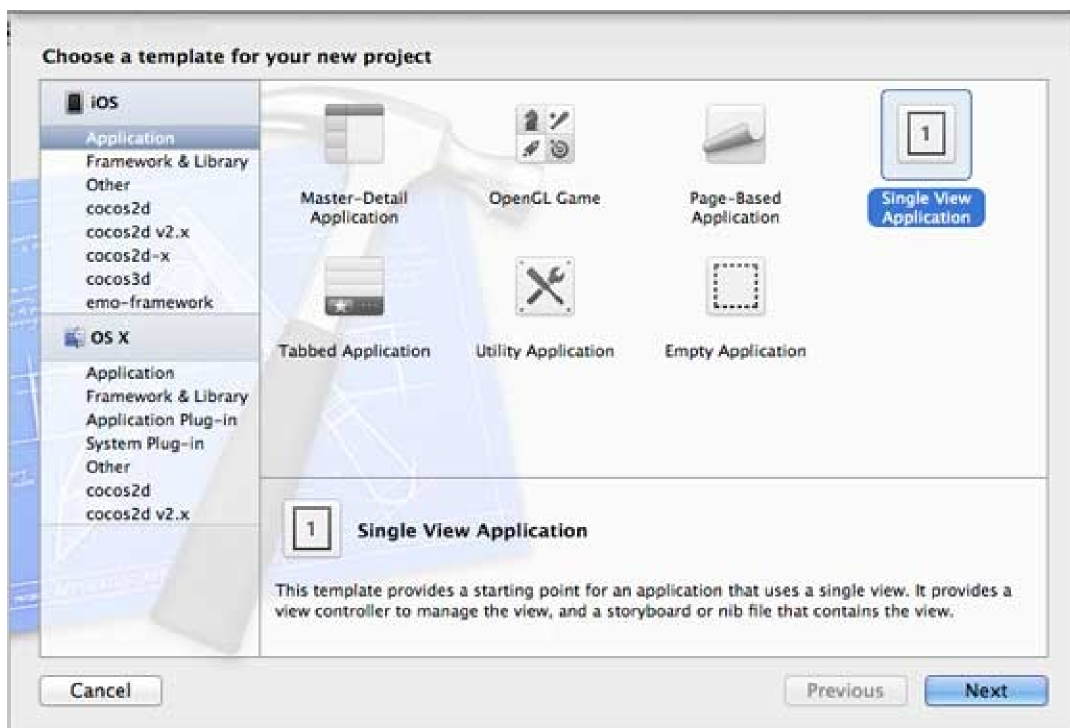
现在让我们来创建一个在iOS模拟器上运行的简单视图应用（空白的应用程序）。

操作步骤如下：

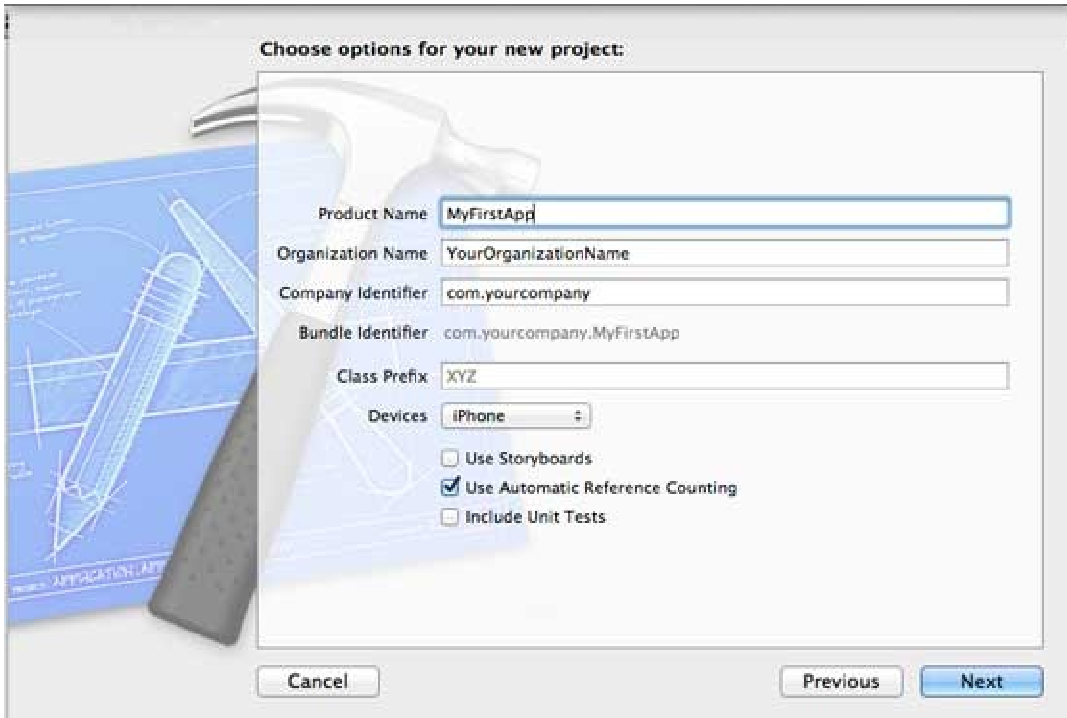
1、打开Xcode并选择创建一个新的Xcode项目。



2. 然后选择单一视图应用程序

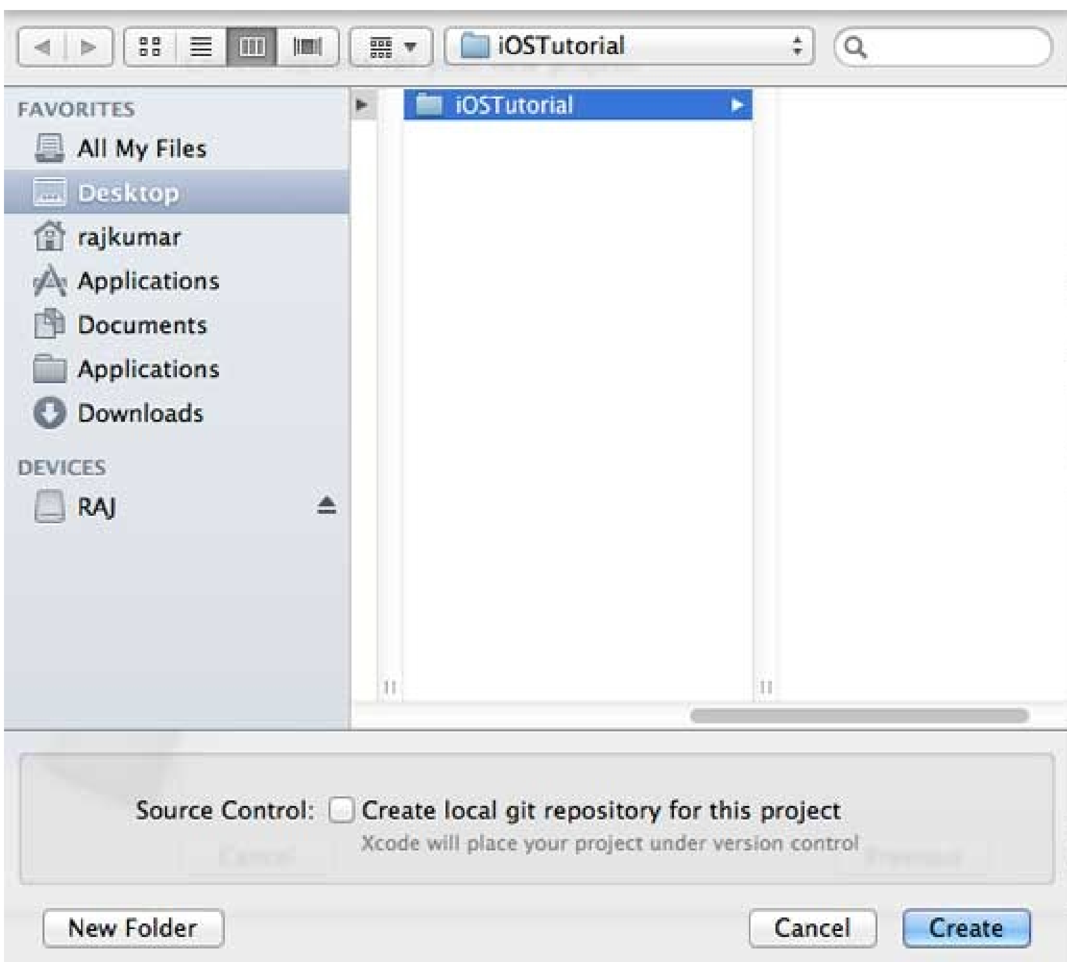


3. 接下来输入产品名称即应用程序名称、组织名称和公司标识符。

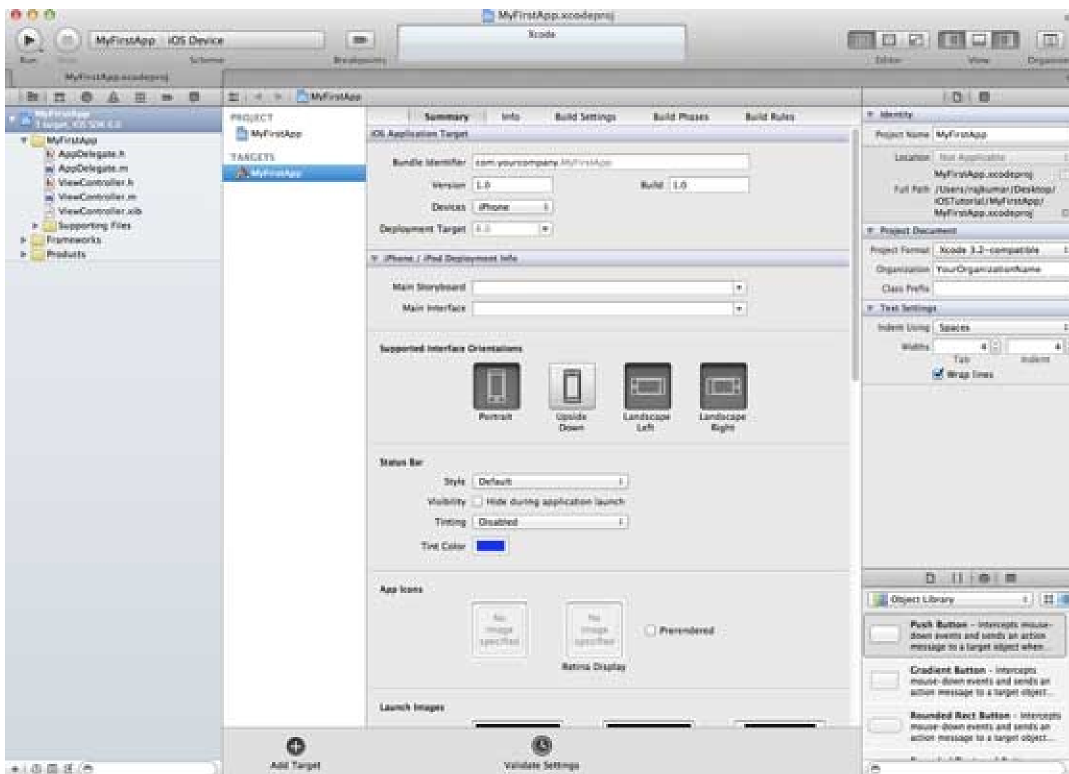


4. 确定已经选择自动应用计数，以自动释放超出范围的资源。单击下一步。

5.选择项目目录并选择创建

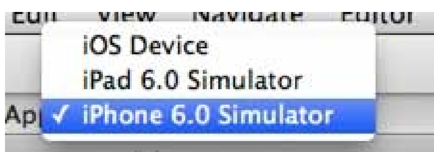


6. 你将看到如下所示的页面

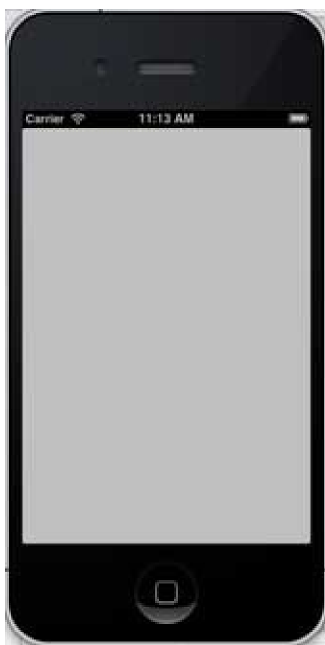


屏幕上方能够设置方向、生成和释放。有一个部署目标，设备支持4.3及以上版本的部署目标，这些不是必须的，现在只要专注于运行该应用程序。

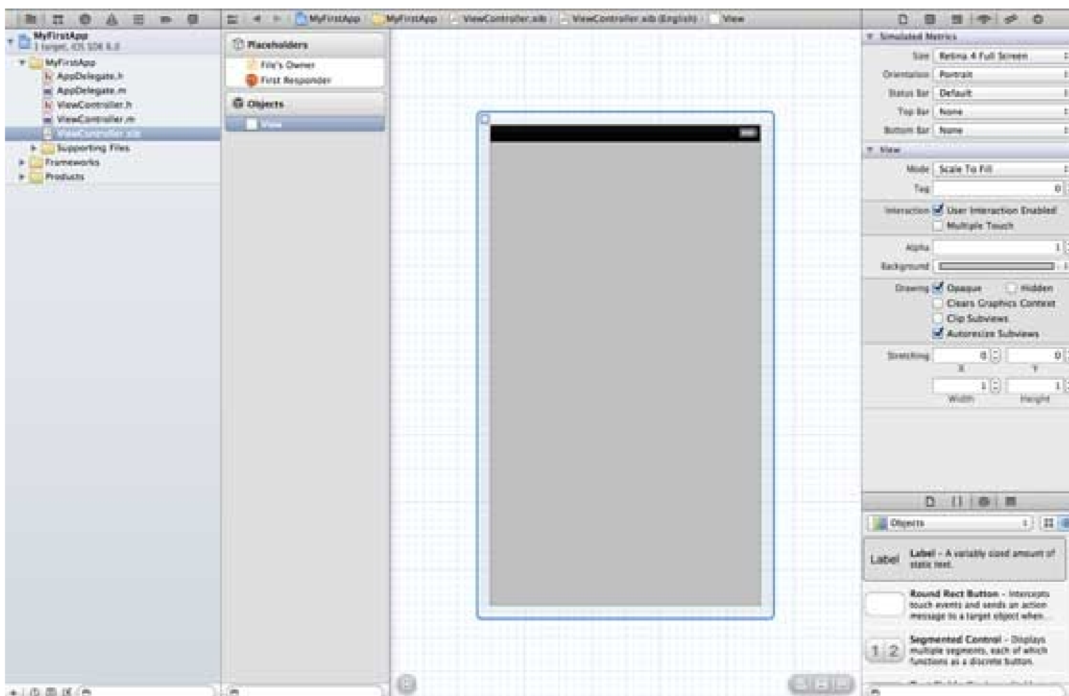
7. 在下拉菜单中选择iPhone Simulator并运行。



8. 成功运行第一个应用程序，将得到的输出，如下所示。



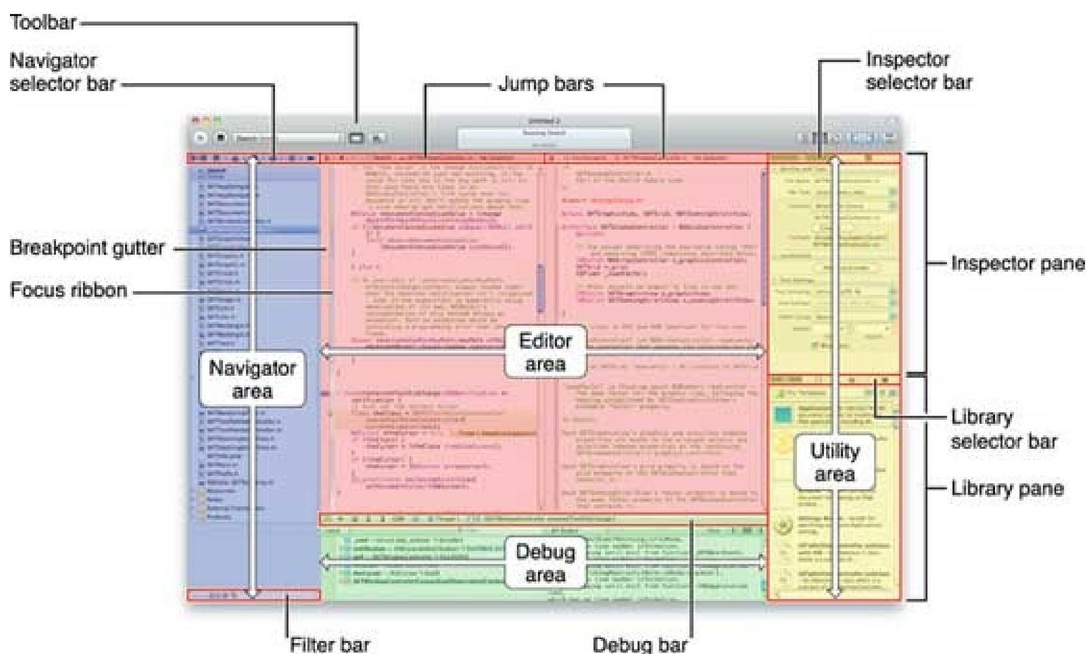
更改背景颜色使之有开始的界面生成器。选择ViewController.xib。在右侧选择背景选项，更改颜色并运行。



在上述项目中，默认情况下，部署目标已设置为iOS6.0且自动布局将被启用。

为确保应用程序能iOS4.3设备上正常运行，我们已经在开始创建应用程序时修改了部署目标，但我们不禁用自动布局，要取消自动布局，我们需要取消选择自动班上复选框在文件查看器的每个nib，也就是xib文件。

Xcode项目IDE的各部分显示如下（苹果Xcode4用户文档）



在上面所示的检查器选择器栏中可以找到文件检查器，且可以取消选择自动布局。当你想要的目标只有iOS6.0的设备时，可以使用自动布局。

当然，也可以使用新功能，如当加注到iOS6时，就可以使用passbook这一功能。现在，以Ios4.3作为部署目标。

深入了解第一款IOS应用程序代码

5个不同文件生成应用程序，如下所示

- AppDelegate.h
- AppDelegate.m
- ViewController.h
- ViewController.m
- ViewController.xib

我们使用单行注释（//）来解释简单代码，重要的项目代码解释在代码下方。

AppDelegate.h

```
// Header File that provides all UI related items.
#import &lt;UIKit/UIKit.h&gt;
 // Forward declaration (Used when class will be defined /imported
@class ViewController;

 // Interface for Appdelegate
@interface AppDelegate : UIResponder &lt;UIApplicationDelegate&gt;
// Property window
@property (strong, nonatomic) UIWindow *window;
 // Property Viewcontroller
@property (strong, nonatomic) ViewController *viewController;
//this marks end of interface
@end
```

代码说明

- AppDelegate调用UIResponder来处理Ios事件。
- 完成UIApplication的命令，提供关键应用程序事件，如启动完毕，终止，等等
- 在iOS设备的屏幕上用UIWindow对象来管理和协调各种视角，它就像其它加载视图的基本视图一样。通常一个应用程序只有一个窗口。
- UIViewController来处理屏幕流

AppDelegate.m

```
// Imports the class Appdelegate's interface
import "AppDelegate.h"

// Imports the viewcontroller to be loaded
#import "ViewController.h"
```

```
// Class definition starts here
@implementation AppDelegate

// Following method intimates us the application launched  successf
- (BOOL)application:(UIApplication *)application
 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:
    [[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[ViewController alloc]
     initWithNibName:@"ViewController" bundle:nil];
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    /* Sent when the application is about to move from active to in
    This can occur for certain types of temporary interruptions
    (such as an incoming phone call or SMS message)
    or when the user quits the application and it begins the transi
    background state. Use this method to pause ongoing tasks, disab
    and throttle down OpenGL ES frame rates. Games should use this
    to pause the game.*/
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    /* Use this method to release shared resources, save user data,
    timers, and store enough application state information    to re
    application to its current state in case it is terminated later
    application supports background execution, this method is calle
    of applicationWillTerminate: when the user quits.*/
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    /* Called as part of the transition from the background to the
    here you can undo many of the changes made on entering the back
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    /* Restart any tasks that were paused (or not yet started) whil
    application was inactive. If the application was previously in
    optionally refresh the user interface.*/
}

- (void)applicationWillTerminate:(UIApplication *)application
{
```

```
    /* Called when the application is about to terminate. Save data
    See also applicationDidEnterBackground:. */
}

@end
```

代码说明

- 此处定义UIApplication。上面定义的所有方法都是应用程序UI调动和不包含任何用户定义的方法。
- UIWindow对象被分配用来保存应用程序分配对象。
- UIController作为窗口初始视图控制器
- 调用makeKeyAndVisible能使窗口可见

ViewController.h

```
#import

// Interface for class ViewController
@interface ViewController : UIViewController

@end
```

代码说明

- ViewController类继承UIViewController，为iOS应用程序提供基本视图管理模型。

ViewController.m

```
#import "ViewController.h"

// Category, an extension of ViewController class
@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

代码说明

- 在这里两种方法实现UIViewController类的基类中定义
- 初始视图加载后调用viewDidLoad中的安装程序
- 在内存警告的情况下调用didReceviveMemoryWarning

简介

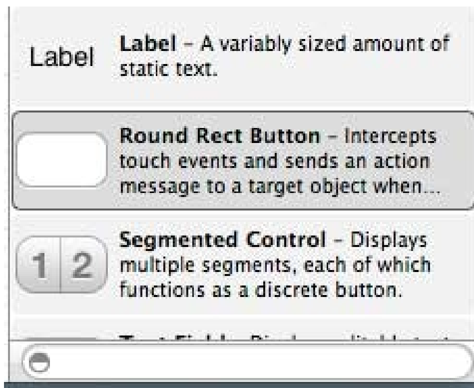
在iOS中，操作（action）和输出口（Outlet）指的是ibActions和ibOutlets，也就是ib接口生成器所在的地方。这些都和UI元素相关，我们将直观的了解他们后探讨如何实现他们。

步骤

1、让我们使用第一款iPhone应用程序。

2、从导航部分中的文件中选择ViewController.xib文件

3、从右手边得窗口下面显示的窗口格库中选择UI元素



4、拖拽UI元素到界面生成器的可视框中

5、添加标签和红色圆形按钮到可视图中



6、在工作区工具栏的右上角找到编辑器选择按钮，如下图所示

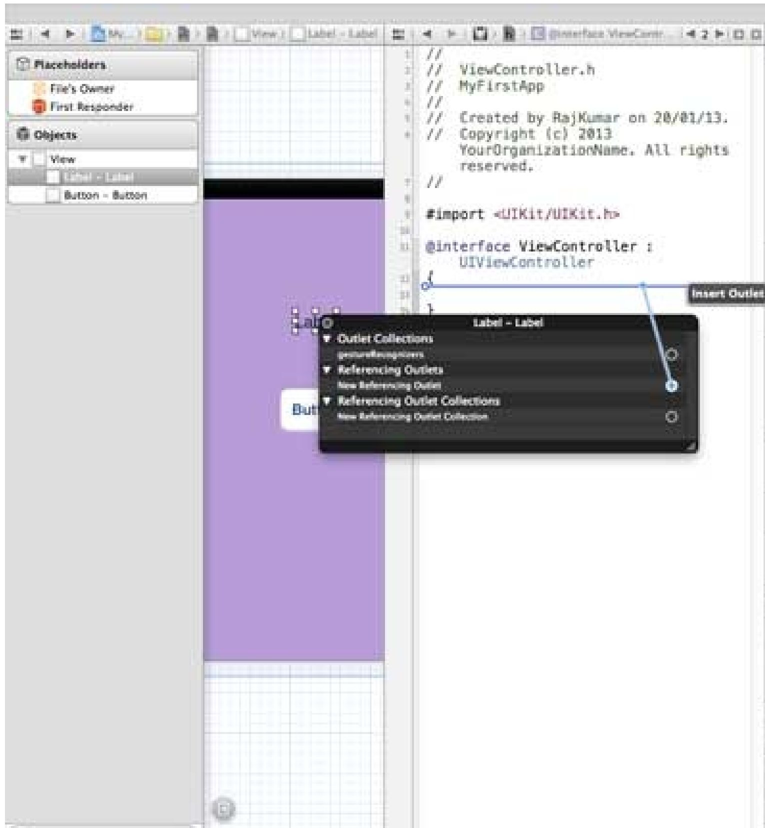


选择编辑器按钮

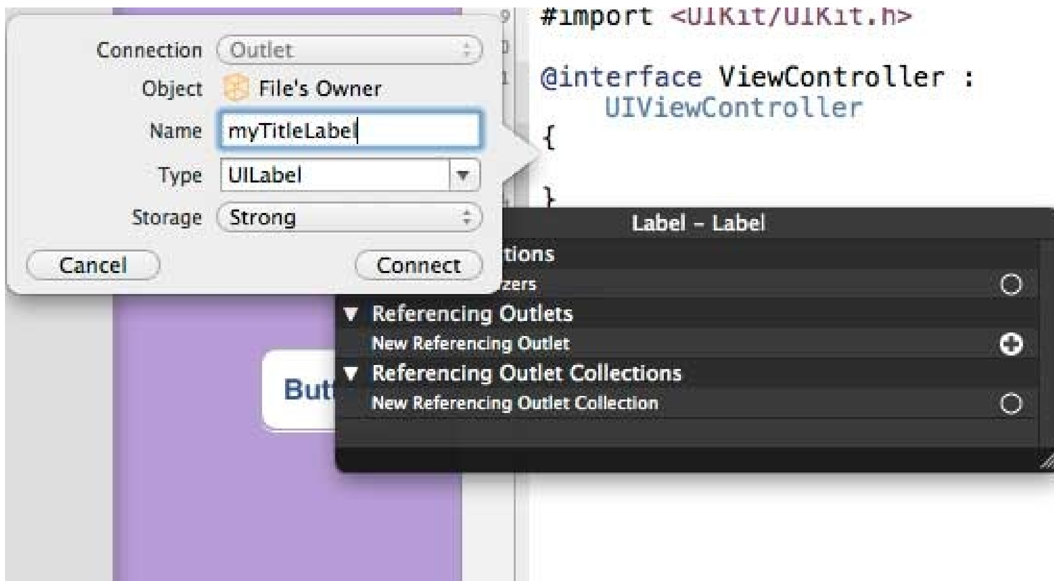


7、编辑器区域中心有两个窗口，ViewController.xib文件和ViewController.h

8、右击标签上的选择按钮，按住并拖动新引用参照，如下所示

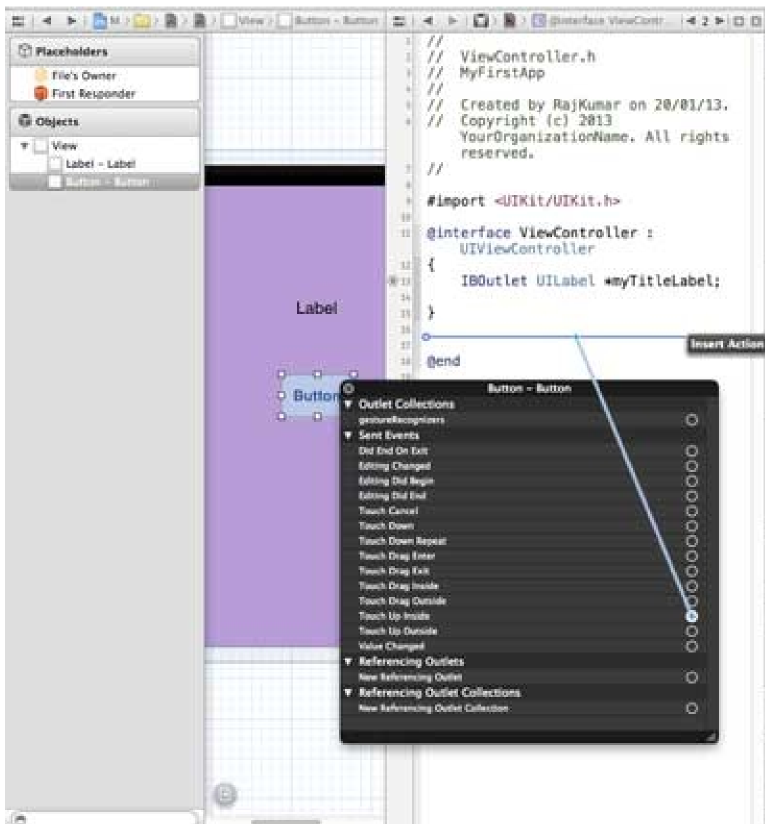


9、现在放在ViewController.h之间的大括号中。也可以放在文件中，如果是这样，必须在做这个之前已经添加了。如下所示

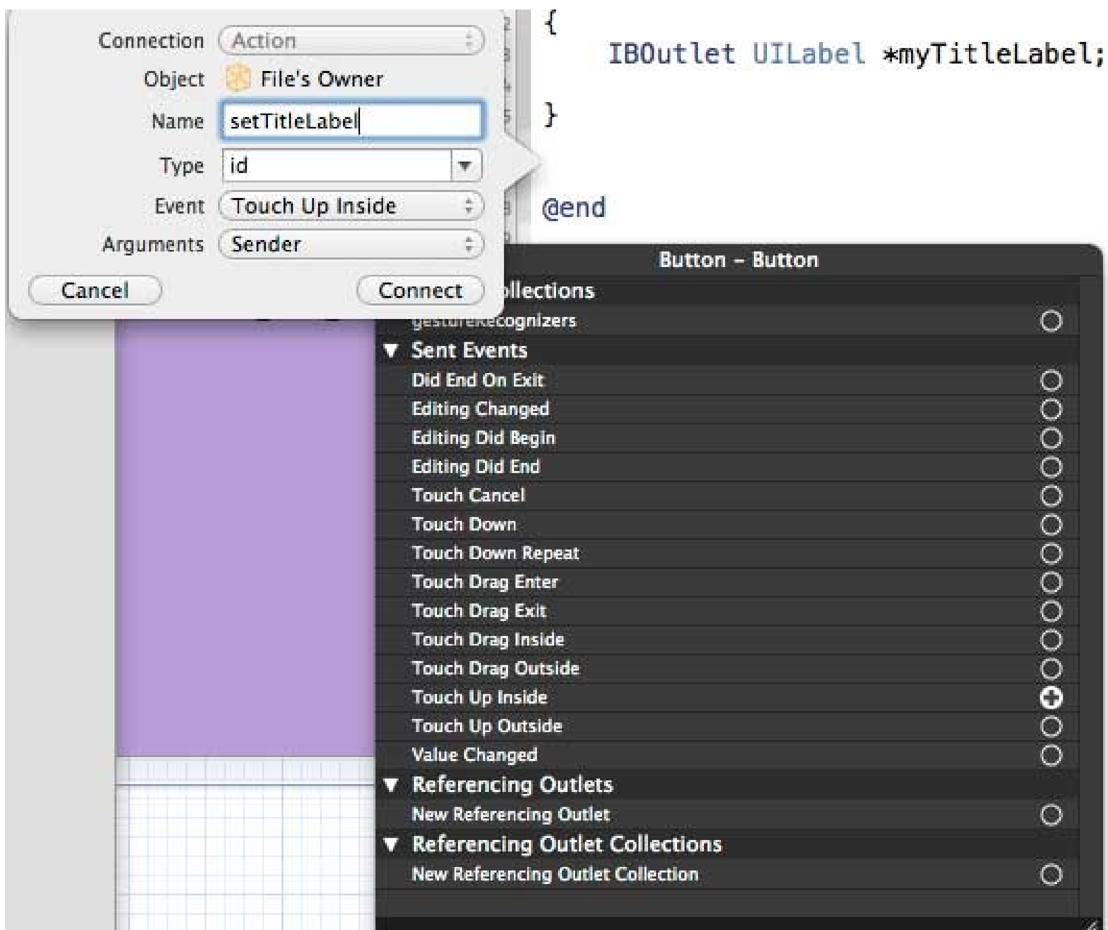


10. 输入输出口（Outlet）的标签名称，这里给出的是myTitleLable。单击链接，完成ibOutlet

11、同样的，添加操作，只需右击倒圆角矩形，选择触摸内心拖动它下方的大括号



12、重新命名为setTitleLable



13、 选择ViewController.m文件，有一种方法，如下所示

```
-(IBAction) setTitleLabel:(id)sender{
}
```

14、在上述的方法内，如下所示，添加一个语句

```
[myTitleLabel setTitleText:@"Hello"];
```

15、选择运行按钮运行该程序，得到如下的输出



16、单击按钮



17.、 创建的参照（outlets）按钮标签已更改为对按钮执行的操作（actions）

18、 由上可知，IBOutlet将创建对UIElement的引用（此处为UILable），同样的IBAction和UIButton通过执行操作和UIButton相链接。

19、 当创建动作时通过选择不同的事件你可以做不同的操作。

委托（Delegates）示例

假设对象A调用B来执行一项操作，操作一旦完成，对象A就必须知道对象B已完成任务且对象A将执行其他必要操作。

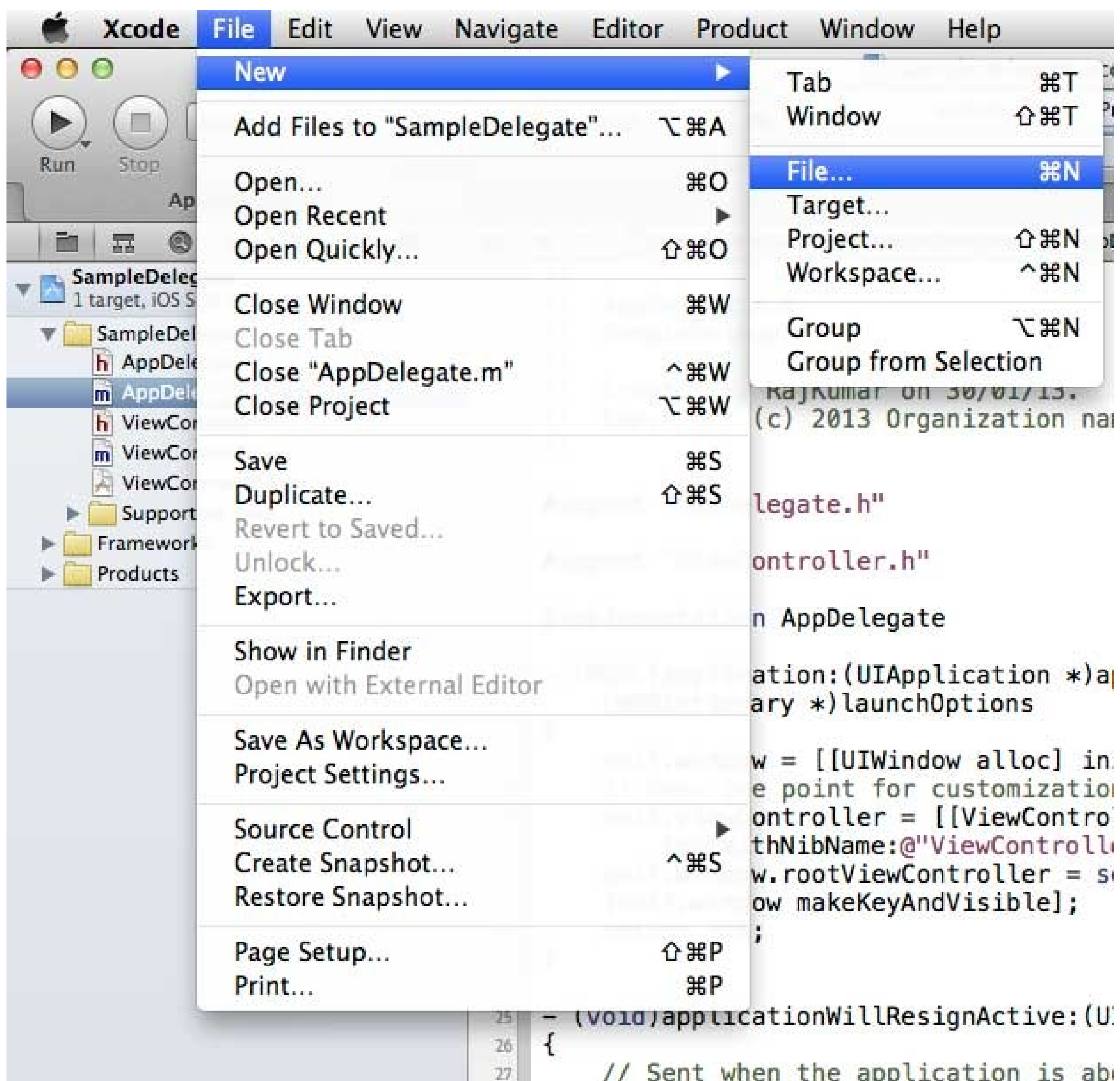
在上面的示例中的关键概念有

- A是B的委托对象
- B引用一个A
- A将实现B的委托方法
- B通过委托方法通知

创建一个委托（Delegates）对象

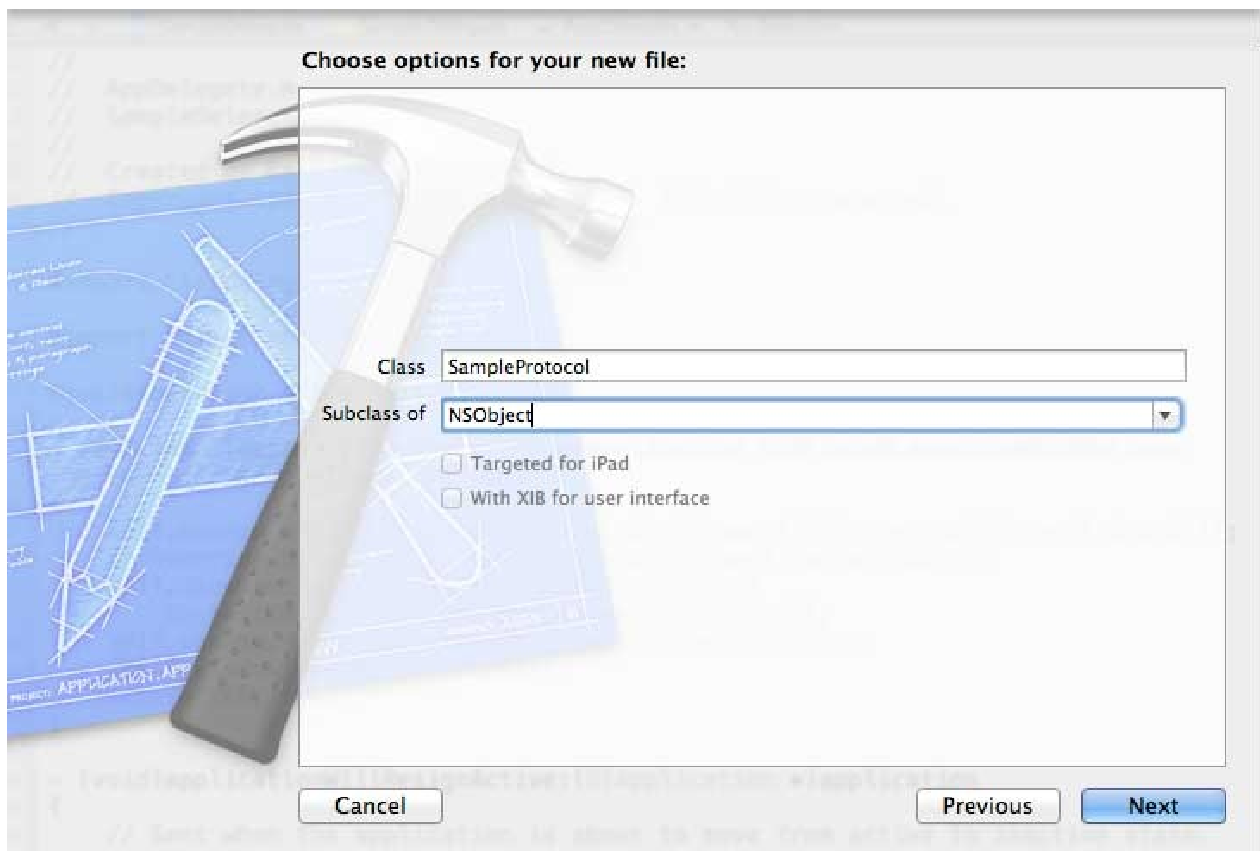
1. 创建一个单一视图的应用程序

2. 然后选择文件 File -> New -> File...



3. 然后选择Objective C单击下一步

4. 将SampleProtocol的子类命名为NSObject，如下所示



5. 然后选择创建

6.向SampleProtocol.h文件夹中添加一种协议，然后更新代码，如下所示：

```
#import &lt;Foundation/Foundation.h&gt;
// Protocol definition starts here
@protocol SampleProtocolDelegate &lt;NSObject&gt;
@required
- (void) processCompleted;
@end
// Protocol Definition ends here
@interface SampleProtocol : NSObject

{
   // Delegate to respond back
   id &lt;SampleProtocolDelegate&gt; _delegate;

}
@property (nonatomic,strong) id delegate;

-(void)startSampleProcess; // Instance method

@end
```

7. Implement the instance method by updating the SampleProtocol.m file as shown below.

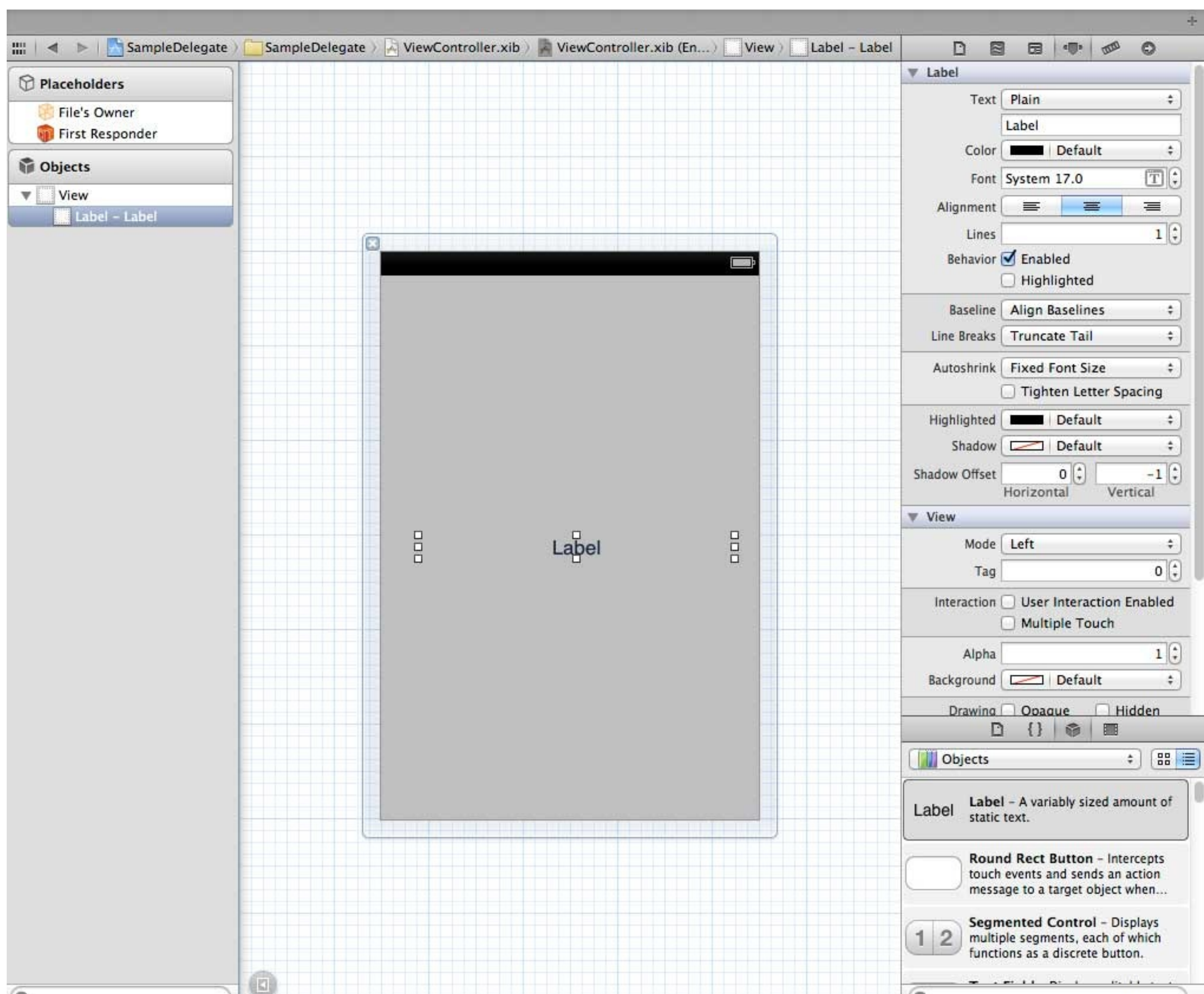
```
#import "SampleProtocol.h"

@implementation SampleProtocol

-(void)startSampleProcess{

    [NSTimer scheduledTimerWithTimeInterval:3.0 target:self.delegat
    selector:@selector(processCompleted) userInfo:nil repeats:NO];
}
@end
```

8. 将标签从对象库拖到UIView，从而在ViewController.xib中添加UILabel，如下所示:



9. 创建一个IBOutlet标签并命名为myLabel，然后按如下所示更新代码并在ViewController.h里显示SampleProtocolDelegate

```
#import &lt;UIKit/UIKit.h&gt;
#import "SampleProtocol.h"

@interface ViewController : UIViewController&lt;SampleProtocolDeleg
{
    IBOutlet UILabel *myLabel;
}
@end
```

10. 完成授权方法，为SampleProtocol创建对象和调用startSampleProcess方法。如下所示，更新ViewController.m文件

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    SampleProtocol *sampleProtocol = [[SampleProtocol alloc]init];
    sampleProtocol.delegate = self;
    [myLabel setText:@"Processing..."];
    [sampleProtocol startSampleProcess];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Sample protocol delegate
-(void)processCompleted{
    [myLabel setText:@"Process Completed"];
}

@end
```

11. 将看到如下所示的输出结果，最初的标签也会继续运行，一旦授权方法被SampleProtocol对象所调用，标签运行程序的代码也会更新。



什么是UI元素？

UI元素是我们应用程序里可以看见的任何可视元素，其中一些元素响应用户的操作，如按钮、文本字段，有其他的丰富内容，如图像、标签等。

如何添加UI元素？

可以在界面生成器的参与下，在代码中添加UI元素。如果需要，我们可以使用他们其中之一。

我们关注的

通过代码，将集中于添加UI元素到应用程序。比较简单而直接的方法是使用界面生成器拖放UI元素。

方法

以下我们通过创建一款简单的IOS应用程序，来解释一些UI元素

步骤

1、在第一款IOS程序里一样，创建一个Viewbased应用程序

2、只更新ViewController.h和ViewController.m文件

3、然后我们将方法添加到ViewController.m文件中来创建UI元素

4、在viewDidLoad方法中调用此方法

5、重要的代码行已经在代码中通过在单行上方标注的方式进行了注释

用户界面元素列表

下面解释具体的UI元素和其相关的功能

具体的UI元素	功能
Text Fields-文本字段	用户界面元素，使用应用程序来获取用户输入
输入类型-TextFields	用户可以通过使用UITextField来赋予键盘输入属性
Buttons-按钮	用于处理用户操作
Label-标签	用于显示静态内容
Toolbar-工具栏	操纵当前视图所显示的东西
Status Bar-状态栏	显示设备的关键信息
Navigation Bar-导航栏	包含一个可以推断的视图控制器，并弹出导航控制器的导航按钮
Tab bar-选项卡栏	一般用于各个子任务、视图或同一视图中的模型之间的切换.
Image View-图像视图	用于显示一个简单的图像序列
Scroll View-滚动视图	用来显示更多屏幕区域的内容
Table View-列表视图	用于在多个行或部分中显示可滚动列表的数据
IOS分割视图(Split View)	用于在详细信息窗格上显示两个窗格与主窗格的控制信息
Text View-文本视图	用于显示滚动列表的文本信息可以被选中和编辑
View Transition -视图切换	各种视图查看之间的切换
Pickers-选择器	用来显示从列表中选择一个特定的数据
Switches-开关	用作禁用和启用操作
IOS滑块(Sliders)	用来允许用户在允许的值范围内选对一个值
IOS警告对话框(Alerts)	用来给用户提供重要的信息
IOS图标(Icons)	它是图像，表示用于行动或描绘与应用程序相关的东西

文本字段的使用

文本字段是一个用户界面元素，通过应用程序来获取用户输入。

一个UITextfield如下所示：

重要的文本字段的属性

- 在没有任何用户输入时，显示占位符
- 正常文本
- 自动更正型
- 键盘类型
- 返回键类型
- 清除按钮模式
- 对齐方式
- 委托

更新xib中的属性

可以在Utility area（实用区域，窗口的右侧）更改xib在属性查看器中的文本字段属性。



文本字段委托

我们可以通过右击 UIElement 界面生成器中设置委托并将它连接到文件的所有者，如下所示：



使用委托的步骤：

- 1.设置委托如上图所示
- 2.添加委托到您的响应类
- 3.执行文本字段代表，重要的文本字段代表如下:

```
- (void)textFieldDidBeginEditing:(UITextField *)textField
```

```
- (void)textFieldDidEndEditing:(UITextField *)textField
```

- 4.正如其名称所暗示，上述两个委托分别叫做编辑的文本字段和结束编辑
- 5. 其他的委托请查看 UITextDelegate Protocol 参考手册。

实例

以下我们使用简单的实例来创建UI元素

ViewController 类将采用UITextFieldDelegate，修改ViewController.h文件，如下所示:

将方法addTextField添加到我们的 ViewController.m 文件

然后在 viewDidLoad 方法中调用此方法

在ViewController.m中更新viewDidLoad，如下所示

```
#import "ViewController.h"
@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our textfield is called
    [self addTextField];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(void)addTextField{
   // This allocates a label
   UILabel *prefixLabel = [[UILabel alloc]initWithFrame:CGRectZero]
   //This sets the label text
   prefixLabel.text =@"## ";
   // This sets the font for the label
   [prefixLabel setFont:[UIFont boldSystemFontOfSize:14]];
   // This fits the frame to size of the text
   [prefixLabel sizeToFit];

   // This allocates the textfield and sets its frame
   UITextField *textField = [[UITextField  alloc] initWithFrame:
   CGRectMake(20, 50, 280, 30)];

   // This sets the border style of the text field
   textField.borderStyle = UITextBorderStyleRoundedRect;
   textField.contentVerticalAlignment =
   UIControlContentVerticalAlignmentCenter;
   [textField setFont:[UIFont boldSystemFontOfSize:12]];

   //Placeholder text is displayed when no text is typed
   textField.placeholder = @"Simple Text field";

   //Prefix label is set as left view and the text starts after tha
   textField.leftView = prefixLabel;
```

```
    //It set when the left prefixLabel to be displayed
    textField.leftViewMode = UITextFieldViewModeAlways;

    // Adds the textField to the view.
    [self.view addSubview:textField];

    // sets the delegate to the current class
    textField.delegate = self;
}

// pragma mark is used for easy access of code in Xcode
#pragma mark - TextField Delegates

// This method is called once we click inside the textField
-(void)textFieldDidBeginEditing:(UITextField *)textField{
    NSLog(@"Text field did begin editing");
}

// This method is called once we complete editing
-(void)textFieldDidEndEditing:(UITextField *)textField{
    NSLog(@"Text field ended editing");
}

// This method enables or disables the processing of return key
-(BOOL) textFieldShouldReturn:(UITextField *)textField{
     [textField resignFirstResponder];
     return YES;
}

- (void)viewDidUnload {
    label = nil;
    [super viewDidUnload];
}

@end
```

运行该应用程序会看到下面的输出

委托调用的方法基于用户操作。要知道调用委托时请参阅控制台输出。

为什么使用不同的输入类型？

键盘输入的类型帮助我们从用户获取必需的输入。

它移除不需要的键，并包括所需的部分。用户可以通过使用 UITextField 的键盘属性设置输入的类型。

- 如：文本字段（textField）。 keyboardType = UIKeyboardTypeDefault

键盘输入类型

输入的类型	描述
UIKeyboardTypeASCIICapable	键盘包括所有标准的 ASCII 字符。
UIKeyboardTypeNumbersAndPunctuation	键盘显示数字和标点。
UIKeyboardTypeURL	键盘的 URL 项优化。
UIKeyboardTypeNumberPad	键盘用于 PIN 输入和显示一个数字键盘。
UIKeyboardTypePhonePad	键盘对输入电话号码进行了优化。
UIKeyboardTypeNamePhonePad	键盘用于输入姓名或电话号码。
UIKeyboardTypeEmailAddress	键盘对输入电子邮件地址的优化。
UIKeyboardTypeDecimalPad	键盘用来输入十进制数字。
UIKeyboardTypeTwitter	键盘对 twitter @ 和 # 符号进行了优化。

添加自定义方法 addTextFieldWithDifferentKeyboard

```
-(void) addTextFieldWithDifferentKeyboard{

   UITextField *textField1= [[UITextField alloc]initWithFrame:
   CGRectMake(20, 50, 280, 30)];
   textField1.delegate = self;
   textField1.borderStyle = UITextBorderStyleRoundedRect;
   textField1.placeholder = @"Default Keyboard";
   [self.view addSubview:textField1];

   UITextField *textField2 = [[UITextField alloc]initWithFrame:
   CGRectMake(20, 100, 280, 30)];
   textField2.delegate = self;
   textField2.borderStyle = UITextBorderStyleRoundedRect;
   textField2.keyboardType = UIKeyboardTypeASCIICapable;
   textField2.placeholder = @"ASCII keyboard";
   [self.view addSubview:textField2];

   UITextField *textField3 = [[UITextField alloc]initWithFrame:
   CGRectMake(20, 150, 280, 30)];
   textField3.delegate = self;
   textField3.borderStyle = UITextBorderStyleRoundedRect;
   textField3.keyboardType = UIKeyboardTypePhonePad;
   textField3.placeholder = @"Phone pad keyboard";
   [self.view addSubview:textField3];

   UITextField *textField4 = [[UITextField alloc]initWithFrame:
   CGRectMake(20, 200, 280, 30)];
   textField4.delegate = self;
   textField4.borderStyle = UITextBorderStyleRoundedRect;
   textField4.keyboardType = UIKeyboardTypeDecimalPad;
   textField4.placeholder = @"Decimal pad keyboard";
   [self.view addSubview:textField4];

   UITextField *textField5= [[UITextField alloc]initWithFrame:
   CGRectMake(20, 250, 280, 30)];
   textField5.delegate = self;
   textField5.borderStyle = UITextBorderStyleRoundedRect;
   textField5.keyboardType = UIKeyboardTypeEmailAddress;
   textField5.placeholder = @"Email keyboard";
   [self.view addSubview:textField5];

   UITextField *textField6= [[UITextField alloc]initWithFrame:
   CGRectMake(20, 300, 280, 30)];
   textField6.delegate = self;
   textField6.borderStyle = UITextBorderStyleRoundedRect;
   textField6.keyboardType = UIKeyboardTypeURL;
   textField6.placeholder = @"URL keyboard";
   [self.view addSubview:textField6];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   //The custom method to create textfield with different keyboard
   [self addTextFieldWithDifferentKeyboard];
   //Do any additional setup after loading the view, typically fron
}
```

输出

现在当我们运行应用程序时我们就会得到下面的输出:

选择不同的文本区域我们将看到不同的键盘。

按钮使用

按钮用于处理用户操作。它截取触摸事件，并将消息发送到目标对象。

圆角矩形按钮

在 xib 中的按钮属性

您可以在Utility area（实用区域，窗口的右侧）的属性检查器的更改 xib 按钮属性。

按钮类型

- UIButtonTypeCustom
- UIButtonTypeRoundedRect
- UIButtonTypeDetailDisclosure
- UIButtonTypeInfoLight
- UIButtonTypeInfoDark
- UIButtonTypeContactAdd

重要的属性

- imageView
- titleLabel

重要的方法

```
+ (id)buttonWithType:(UIButtonType)buttonType
```

```
- (UIImage *)backgroundImageForState:(UIControlState)state
```

```
- (UIImage *)imageForState:(UIControlState)state
```

```
- (void)setTitle:(NSString *)title forState:(UIControlState)state
```

```
- (void)addTarget:(id)target action:(SEL)action forControlEvents: (
```

添加自定义方法 addDifferentTypesOfButton

```
-(void)addDifferentTypesOfButton
{
    // A rounded Rect button created by using class method
    UIButton *roundRectButton = [UIButton buttonWithType:
    UIButtonTypeRoundedRect];
    [roundRectButton setFrame:CGRectMake(60, 50, 200, 40)];
    // sets title for the button
    [roundRectButton setTitle:@"Rounded Rect Button" forState:
    UIControlStateNormal];
    [self.view addSubview:roundRectButton];

    UIButton *customButton = [UIButton buttonWithType: UIButtonType
    [customButton setBackgroundColor: [UIColor lightGrayColor]];
    [customButton setTitleColor:[UIColor blackColor] forState:
    UIControlStateHighlighted];
    //sets background image for normal state
    [customButton setBackgroundImage:[UIImage imageNamed:
    @"Button_Default.png"]
    forState:UIControlStateNormal];
    //sets background image for highlighted state
    [customButton setBackgroundImage:[UIImage imageNamed:
    @"Button_Highlighted.png"]
    forState:UIControlStateHighlighted];
    [customButton setFrame:CGRectMake(60, 100, 200, 40)];
    [customButton setTitle:@"Custom Button" forState:UIControlState
    [self.view addSubview:customButton];

    UIButton *detailDisclosureButton = [UIButton buttonWithType:
    UIButtonTypeDetailDisclosure];
    [detailDisclosureButton setFrame:CGRectMake(60, 150, 200, 40)]
    [detailDisclosureButton setTitle:@"Detail disclosure" forState
    UIControlStateNormal];
    [self.view addSubview:detailDisclosureButton];

    UIButton *contactButton = [UIButton buttonWithType:
    UIButtonTypeContactAdd];
    [contactButton setFrame:CGRectMake(60, 200, 200, 40)];
    [self.view addSubview:contactButton];

    UIButton *infoDarkButton = [UIButton buttonWithType:
    UIButtonTypeInfoDark];
    [infoDarkButton setFrame:CGRectMake(60, 250, 200, 40)];
    [self.view addSubview:infoDarkButton];

    UIButton *infoLightButton = [UIButton buttonWithType:
    UIButtonTypeInfoLight];
    [infoLightButton setFrame:CGRectMake(60, 300, 200, 40)];
    [self.view addSubview:infoLightButton];
}
```

注意：

我们将命名为"Button_Default.png"和"Button_Highlighted.png"的个图像添加到我们的项目，可以通过将图像拖到列出了我们的项目文件的导航区域来完成。

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our different types of button is
    [self addDifferentTypesOfButton];
    //Do any additional setup after loading the view, typically fro
}
```

输出

现在当我们运行应用程序时我们就会得到下面的输出：



标签的使用

标签用于显示静态内容，包括单独的一行或多行。

重要的属性

- textAlignment
- textColor
- text
- numberOflines
- lineBreakMode

添加自定义方法 addLabel

```
-(void)addLabel{
    UILabel *aLabel = [[UILabel alloc]initWithFrame:
    CGRectMake(20, 200, 280, 80)];
    aLabel.numberOfLines = 0;
    aLabel.textColor = [UIColor blueColor];
    aLabel.backgroundColor = [UIColor clearColor];
    aLabel.textAlignment = UITextAlignmentCenter;
    aLabel.text = @"This is a sample text\n of multiple lines.
    here number of lines is not limited.";
    [self.view addSubview:aLabel];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our label is called
    [self addLabel];
    // Do any additional setup after loading the view, typically fr
}
```

输出

运行应用程序，就会得到下面的输出:



工具栏的使用

我们可以使用工具栏修改视图元素。

如，邮件应用程序里的收件箱栏中有删除、分享、答复等等。如下所示：



重要的属性

- barStyle
- items

添加自定义方法 addToolbar

```
-(void)addToolbar
{
    UIBarButtonItem *spaceItem = [[UIBarButtonItem alloc]
    initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace
    target:nil action:nil];
    UIBarButtonItem *customItem1 = [[UIBarButtonItem alloc]
    initWithTitle:@"Tool1" style:UIBarButtonItemStyleBordered
    target:self action:@selector(toolBarItem1:)];
    UIBarButtonItem *customItem2 = [[UIBarButtonItem alloc]
    initWithTitle:@"Tool2" style:UIBarButtonItemStyleDone
    target:self action:@selector(toolBarItem2:)];
    NSArray *toolbarItems = [NSArray arrayWithObjects:
    customItem1,spaceItem, customItem2, nil];
    UIToolbar *toolbar = [[UIToolbar alloc]initWithFrame:
    CGRectMake(0, 366+54, 320, 50)];
    [toolbar setBarStyle:UIBarStyleBlackOpaque];
    [self.view addSubview:toolbar];
    [toolbar setItems:toolbarItems];
}
```

为了解所执行的操作我们在我们的ViewController.xib中添加UILabel Iboutlet并为UILabel 创建命名为标签的IBoutlet。

我们还需要添加两个方法来执行，如下所示的工具栏项的操作:

```
-(IBAction)toolBarItem1:(id)sender{
    [label setText:@"Tool 1 Selected"];
}

-(IBAction)toolBarItem2:(id)sender{
    [label setText:@"Tool 2 Selected"];
}
```

在ViewController.m中更新 viewDidLoad，如下所示:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // The method hideStatusbar called after 2 seconds
    [self addToolbar];
    // Do any additional setup after loading the view, typically fı
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出。



单击我们得到的 tool1 和 tool2 栏按钮



状态栏的使用

状态栏显示设备的关键信息。

- 设备模型或网络提供商
- 网络信号强度
- 电池使用量
- 时间

状态栏如下所示：



隐藏状态栏的方法

```
[[UIApplication sharedApplication] setStatusBarHidden:YES];
```

另一种隐藏状态栏的方法

我们还可以通过添加行，并在info.plist 的帮助下选择 UIStatusBarHidden 隐藏状态栏，并使其值为否（NO）。

在类中添加自定义方法 hideStatusbar

它隐藏状态栏进行动画处理，并也调整我们认为占据状态栏空间的大小。

```
-(void)hideStatusbar{
    [[UIApplication sharedApplication] setStatusBarHidden:YES
    withAnimation:UIStatusBarAnimationFade];
    [UIView beginAnimations:@"Statusbar hide" context:nil];
    [UIView setAnimationDuration:0.5];
    [self.view setFrame:CGRectMake(0, 0, 320, 480)];
    [UIView commitAnimations];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // The method hideStatusbar called after 2 seconds
    [self performSelector:@selector(hideStatusbar)
    withObject:nil afterDelay:2.0];
    // Do any additional setup after loading the view, typically fr
}
```

初始输出以及2秒后输出

IOS导航栏的使用

导航栏包含导航控制器的导航的按钮。在导航栏中的标题是当前视图控制器的标题。

示例代码和步骤

1.创视图应用程序

1. 现在，选择应用程序 Delegate.h ，添加导航控制器的属性，如下所示:

```
#import <UIKit/UIKit.h>

@class ViewController;

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@property (strong, nonatomic) ViewController *viewController;

@property (strong, nonatomic) UINavigationController *navControlle

@end
```

3. 更新应用程序: didFinishLaunchingWithOptions:方法，在AppDelegate.m文件分配的导航控制器，并使其成为窗口的根视图控制器，如下所示：

```
- (BOOL)application:(UIApplication *)application didFinishLaunching
{
    self.window = [[UIWindow alloc] initWithFrame:
    [[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[ViewController alloc]
    initWithNibName:@"ViewController" bundle:nil];
    //Navigation controller init with ViewController as root
    UINavigationController *navController = [[UINavigationControlle
    initWithRootViewController:self.viewController];
    self.window.rootViewController = navController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

4.现在，通过选择**File** -> **New** ->**File...** -> Objective C Class 添加新的类文件 TempViewController，然后将类命名 TempViewController 与 UIViewController 的子类。

5.在ViewController.h中添加navButon，如下所示

```
// ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    UIButton *navButton;
}
@end
```

6.现在添加方法addNavigationBarItem并在viewDidLoad调用方法

7. 为导航项创建方法

1. 我们还需要创建另一种方法到另一视图控制器 TempViewController。
2. 更新后的ViewController.m，如下所示:

```
// ViewController.m
#import "ViewController.h"
#import "TempViewController.h"
@interface ViewController ()

@end
@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self addNavigationBarButton];
    //Do any additional setup after loading the view, typically fro
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(IBAction)pushNewView:(id)sender{
   TempViewController *tempVC =[[TempViewController alloc]
   initWithNibName:@"TempViewController" bundle:nil];
   [self.navigationController pushViewController:tempVC animated:YE
}

-(IBAction)myButtonClicked:(id)sender{
   // toggle hidden state for navButton
   [navButton setHidden:!nav.hidden];
}

-(void)addNavigationBarButton{
   UIBarButtonItem *myNavBtn = [[UIBarButtonItem alloc] initWithTit
   @"MyButton" style:UIBarButtonItemStyleBordered target:
   self action:@selector(myButtonClicked:)];

   [self.navigationController.navigationBar setBarStyle:UIBarStyleE
   [self.navigationItem setRightBarButtonItem:myNavBtn];

   // create a navigation push button that is initially hidden
   navButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
   [navButton setFrame:CGRectMake(60, 50, 200, 40)];
   [navButton setTitle:@"Push Navigation" forState:UIControlStateNo
   [navButton addTarget:self action:@selector(pushNewView:)
   forControlEvents:UIControlEventTouchUpInside];
   [self.view addSubview:navButton];
   [navButton setHidden:YES];
}
@end
```

1. 现在当我们运行应用程序时我们就会得到下面的输出



1. 单击 MyButton 导航按钮，切换导航按钮可见性
2. 单击导航按钮，显示另一个视图控制器，如下所示



IOS选项卡栏的使用

它一般用于在同一视图中各个子任务、 视图或的模型之间切换。

选项卡栏的示例如下所示：



重要的属性

- backgroundImage
- items
- selectedItem

示例代码和步骤

1. 创建一个新的项目，选择 **Tabbed Application** 替代视图应用程序 ,点击下一步, 输入项目名称和选择 **create**.

1. 这里默认创建两个视图控制器和标签栏添加到我们的应用程序。

3. **AppDelegate.m didFinishLaunchingWithOptions**方法如下：

```
- (BOOL)application:(UIApplication *)application didFinishLaunching
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainSc
    bounds]];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[FirstViewController alloc
    initWithNibName:@"FirstViewController" bundle:nil];
    UIViewController *viewController2 = [[SecondViewController allo
    initWithNibName:@"SecondViewController" bundle:nil];
    self.tabBarController = [[UITabBarController alloc] init];
    self.tabBarController.viewControllers = @[viewController1,
    viewController2];
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

1. 两个视图控制器被用来分配作为选项卡栏控制器的视图控制器
2. 运行应用程序,得到如下结果：



图像视图的使用

图像视图用于显示单个图像或动画序列的图像。

重要的属性

- image
- highlightedImage
- userInteractionEnabled
- animationImages
- animationRepeatCount

重要的方法

```
- (id)initWithImage:(UIImage *)image
```

```
- (id)initWithImage:(UIImage *)image highlightedImage:
  (UIImage *)highlightedImage
```

```
- (void)startAnimating
```

```
- (void)stopAnimating
```

添加自定义方法 **addImageView**

```
-(void)addImageView{
    UIImageView *imgview = [[UIImageView alloc]
    initWithFrame:CGRectMake(10, 10, 300, 400)];
    [imgview setImage:[UIImage imageNamed:@"AppleUSA1.jpg"]];
    [imgview setContentMode:UIViewContentModeScaleAspectFit];
    [self.view addSubview:imgview];
}
```

添加另一个自定义方法 **addImageViewWithAnimation**

这种方法解释了如何对imageView 中的图像进行动画处理

```
-(void)addImageViewWithAnimation{
    UIImageView *imgview = [[UIImageView alloc]
    initWithFrame:CGRectMake(10, 10, 300, 400)];
    // set an animation
    imgview.animationImages = [NSArray arrayWithObjects:
    [UIImage imageNamed:@"AppleUSA1.jpg"],
    [UIImage imageNamed:@"AppleUSA2.jpg"], nil];
    imgview.animationDuration = 4.0;
    imgview.contentMode = UIViewContentModeCenter;
    [imgview startAnimating];
    [self.view addSubview:imgview];
}
```

注意： 我们必须添加命名为"AppleUSA1.jpg"和"AppleUSA2.jpg"到我们的项目，可以通过将图像拖到我们导航区域，其中列出了我们的项目文件所做的图像。

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addImageView];
}
```

滚动视图的使用

如果内容超出屏幕的大小就会使用到滚动视图来显示隐藏的部分。

它可以包含所有的其他用户界面元素 如图像视图、 标签、 文本视图甚至另一个滚动视图。

重要的属性

- contentSize
- contentInset
- contentOffset
- delegate

重要的方法

```
- (void)scrollRectToVisible:(CGRect)rect animated:(BOOL)animated
```

```
- (void)setContentOffset:(CGPoint)contentOffset animated:(BOOL)anin
```

重要的委托方法

在ViewController.h中，加入<uiscrollviewdelegate>滚动视图和声明滚动视图让类符合委托协议，如下所示:</uiscrollviewdelegate>

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIScrollViewDelegate>
{
    UIScrollView *myScrollView;
}

@end
```

添加自定义方法 addScrollView

```
-(void)addScrollView{
    myScrollView = [[UIScrollView alloc]initWithFrame:
    CGRectMake(20, 20, 280, 420)];
    myScrollView.accessibilityActivationPoint = CGPointMake(100, 1(
    imgView = [[UIImageView alloc]initWithImage:
    [UIImage imageNamed:@"AppleUSA.jpg"]];
    [myScrollView addSubview:imgView];
    myScrollView.minimumZoomScale = 0.5;
    myScrollView.maximumZoomScale = 3;
    myScrollView.contentSize = CGSizeMake(imgView.frame.size.width,
    imgView.frame.size.height);
    myScrollView.delegate = self;
    [self.view addSubview:myScrollView];
}
```

注意： 我们必须添加一个命名为"AppleUSA1.jpg"到我们的项目，可以通过将图像拖到我们导航区域，其中列出了我们的项目文件所做的图像。图像应高于设备的高度。

ViewController.m中实现scrollView 委托

```
 -(UIView *)viewForZoomingInScrollView:(UIScrollView *)scrollView{
    return imgView;
}
-(void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView{
    NSLog(@"Did end decelerating");
}
-(void)scrollViewDidScroll:(UIScrollView *)scrollView{
//    NSLog(@"Did scroll");
}
-(void)scrollViewDidEndDragging:(UIScrollView *)scrollView
  willDecelerate:(BOOL)decelerate{
    NSLog(@"Did end dragging");
}
-(void)scrollViewWillBeginDecelerating:(UIScrollView *)scrollView{
    NSLog(@"Did begin decelerating");
}
-(void)scrollViewWillBeginDragging:(UIScrollView *)scrollView{
    NSLog(@"Did begin dragging");
}
```

在 **ViewController.m** 中更新 **viewDidLoad**，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addScrollView];
   //Do any additional setup after loading the view, typically fro
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出。一旦滚动滚动视图，将能够查看图像的其余部分:



表格视图的使用

IOS表格视图由单元格 （一般可重复使用） 组成，用于显示垂直滚动的视图。

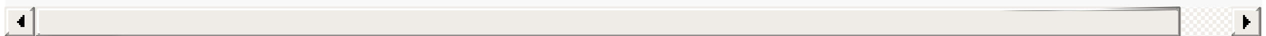
在iOS 中,表格视图用于显示数据列表,如联系人、待办事项或购物项列表。

重要的属性

- delegate
- dataSource
- rowHeight
- sectionFooterHeight
- sectionHeaderHeight
- separatorColor
- tableHeaderView
- tableFooterView

重要的方法

```
- (UITableViewCell *)cellForRowAtIndexPath:(NSIndexPath *)indexPath
```



```
- (void)deleteRowsAtIndexPaths:(NSArray *)indexPaths
  withRowAnimation:(UITableViewRowAnimation)animation
```

```
- (id)dequeueReusableCellWithIdentifier:(NSString *)identifier
```

```
- (id)dequeueReusableCellWithIdentifier:(NSString *)identifier
  forIndexPath:(NSIndexPath *)indexPath
```

```
- (void)reloadData
```

```
- (void)reloadRowsAtIndexPaths:(NSArray *)indexPaths
  withRowAnimation:(UITableViewRowAnimation)animation
```

```
- (NSArray *)visibleCells
```

示例代码和步骤

1.在ViewController.xib中添加表格视图，如下所示



2. 通过右键单击并选择数据源和委托将委托和数据源设定到"File's Owner（文件的所有者）"。设置数据源如下所示



3.为表格视图创建IBOutlet的并将其命名为myTableView。如以下图片中所示





4. 为拥有数据，添加一个NSMutableArray使其能够在列表格视图中显示

5.ViewController应采用的UITableViewDataSource和UITableViewDelegate协议。ViewController.h代码如下所示

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UITableViewDataSource,
UITableViewDelegate>
{

    IBOutlet UITableView *myTableView;
    NSMutableArray *myData;
}

@end
```

6.执行所需的表格视图委托和数据源的方法。更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // table view data is being set here
    myData = [[NSMutableArray alloc]initWithObjects:
    @"Data 1 in array",@"Data 2 in array",@"Data 3 in array",
    @"Data 4 in array",@"Data 5 in array",@"Data 5 in array",
    @"Data 6 in array",@"Data 7 in array",@"Data 8 in array",
    @"Data 9 in array", nil];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
#pragma mark - Table View Data source
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSecti
  (NSInteger)section{
    return [myData count]/2;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowA
  (NSIndexPath *)indexPath{
    static NSString *cellIdentifier = @"cellID";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdent
    cellIdentifier];
    if (cell == nil) {
        cell = [[UITableViewCell alloc]initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:cellIdentifier]
    }
    NSString *stringForCell;
    if (indexPath.section == 0) {
        stringForCell= [myData objectAtIndex:indexPath.row];

    }
    else if (indexPath.section == 1){
        stringForCell= [myData objectAtIndex:indexPath.row+ [myData

    }
    [cell.textLabel setText:stringForCell];
    return cell;
}

// Default is 1 if not implemented
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{
    return 2;
}

- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSe
  (NSInteger)section{
    NSString *headerTitle;
    if (section==0) {
        headerTitle = @"Section 1 Header";
    }
    else{
        headerTitle = @"Section 2 Header";

    }
    return headerTitle;
}
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSe
  (NSInteger)section{
    NSString *footerTitle;
    if (section==0) {
        footerTitle = @"Section 1 Footer";
    }
```

```
    else{
        footerTitle = @"Section 2 Footer";

    }
    return footerTitle;
}

#pragma mark - TableView delegate

-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
 (NSIndexPath *)indexPath{
    [tableView deselectRowAtIndexPath:indexPath animated:YES];
    UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexP
    NSLog(@"Section:%d Row:%d selected and its data is %@",
    indexPath.section,indexPath.row,cell.textLabel.text);
}

@end
```

7.现在当我们运行应用程序时我们就会得到下面的输出



分割视图的使用

分割视图是 iPad 的特定视图控制器用于管理两个视图控制器，在左侧是一个主控制器，其右侧是一个详细信息视图控制器。 重要的属性

- delegate
- viewControllers

示例代码和步骤

1.创建一个新项目，选择Master Detail Application并单击下一步，输入项目名称，然后选择创建。

2.简单的分割视图控制器与母版中的表视图是默认创建的。

3.在这里我们为我们创建的下列文件。

- AppDelegate.h
- AppDelegate.m
- DetailViewController.h
- DetailViewController.m
- DetailViewController.xib
- MasterViewController.h
- MasterViewController.m
- MasterViewController.xib

4. AppDelegate.h文件如下所示

```
#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@property (strong, nonatomic) UISplitViewController *splitViewContr

@end
```

5.在AppDelegate.m中的didFinishLaunchingWithOptions方法，如下所示

```
 - (BOOL)application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
   self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScr
   bounds]];
    // Override point for customization after application launch.
    MasterViewController *masterViewController = [[MasterViewContro
    alloc] initWithNibName:@"MasterViewController" bundle:nil];
    UINavigationController *masterNavigationController =
    [[UINavigationController alloc] initWithRootViewController:
    masterViewController];

    DetailViewController *detailViewController =
    [[DetailViewController alloc] initWithNibName:@"DetailViewContr
    bundle:nil];
    UINavigationController *detailNavigationController =
    [[UINavigationController alloc] initWithRootViewController:
    detailViewController];

    masterViewController.detailViewController = detailViewControlle

    self.splitViewController = [[UISplitViewController alloc] init]
    self.splitViewController.delegate = detailViewController;
    self.splitViewController.viewControllers =
    @[masterNavigationController, detailNavigationController];
    self.window.rootViewController = self.splitViewController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

6. MasterViewController.h，如下所示

```
#import <UIKit/UIKit.h>

@class DetailViewController;

@interface MasterViewController : UITableViewController

@property (strong, nonatomic) DetailViewController *detailViewContr

@end
```

7. MasterViewController.m，如下所示

```
#import "MasterViewController.h"

#import "DetailViewController.h"

@interface MasterViewController () {
    NSMutableArray *_objects;
}
@end

@implementation MasterViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
  nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNi
    if (self) {
        self.title = NSLocalizedString(@"Master", @"Master");
        self.clearsSelectionOnViewWillAppear = NO;
        self.contentSizeForViewInPopover = CGSizeMake(320.0, 600.0
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.navigationItem.leftBarButtonItem = self.editButtonItem;

    UIBarButtonItem *addButton = [[UIBarButtonItem alloc]
    initWithBarButtonSystemItem: UIBarButtonSystemItemAdd
    target:self action:@selector(insertNewObject:)];
    self.navigationItem.rightBarButtonItem = addButton;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
```

```
}

- (void)insertNewObject:(id)sender
{
    if (!_objects) {
        _objects = [[NSMutableArray alloc] init];
    }
    [_objects insertObject:[NSDate date] atIndex:0];
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSecti
    [self.tableView insertRowsAtIndexPaths:@[indexPath] withRowAnim
    UITableViewRowAnimationAutomatic];
}

#pragma mark - Table View

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSecti
  (NSInteger)section
{
    return _objects.count;
}

// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowA
  (NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdent
    CellIdentifier];
    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]
    }

    NSDate *object = _objects[indexPath.row];
    cell.textLabel.text = [object description];
    return cell;
}

- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    // Return NO if you do not want the specified item to be editab
    return YES;
}

- (void)tableView:(UITableView *)tableView commitEditingStyle:
  (UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:
```

```
  (NSIndexPath *)indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [_objects removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnima
        UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert i
        //the array, and add a new row to the table view.
    }
}

/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:
  (NSIndexPath *) fromIndexPath toIndexPath:(NSIndexPath *)toIndexP
{
}
*/

/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    // Return NO if you do not want the item to be re-orderable.
    return YES;
}
*/

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    NSDate *object = _objects[indexPath.row];
    self.detailViewController.detailItem = object;
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    [formatter setDateFormat: @"yyyy-MM-dd HH:mm:ss zzz"];
    NSString *stringFromDate = [formatter stringFromDate:object];
    self.detailViewController.detailDescriptionLabel.text = stringF
}

@end
```

8. DetailViewController.h， 如下所示

```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController
<UISplitViewControllerDelegate>

@property (strong, nonatomic) id detailItem;

@property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabe
@end
```

9. DetailViewController.m， 如下所示

```
#import "DetailViewController.h"

@interface DetailViewController ()
@property (strong, nonatomic) UIPopoverController *masterPopoverCon
- (void)configureView;
@end

@implementation DetailViewController

#pragma mark - Managing the detail item

- (void)setDetailItem:(id)newDetailItem
{
    if (_detailItem != newDetailItem) {
        _detailItem = newDetailItem;

        // Update the view.
        [self configureView];
    }

    if (self.masterPopoverController != nil) {
        [self.masterPopoverController dismissPopoverAnimated:YES];
    }
}

- (void)configureView
{
    // Update the user interface for the detail item.

    if (self.detailItem) {
        self.detailDescriptionLabel.text = [self.detailItem descrip
    }
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self configureView];
```

```
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:
  (NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNi
    if (self) {
        self.title = NSLocalizedString(@"Detail", @"Detail");
    }
    return self;
}

#pragma mark - Split view

- (void)splitViewController:(UISplitViewController *)splitControlle
  willHideViewController:(UIViewController *)viewController withBar
  (UIBarButtonItem *)barButtonItem forPopoverController:
  (UIPopoverController *)popoverController
{
    barButtonItem.title = NSLocalizedString(@"Master", @"Master");
    [self.navigationItem setLeftBarButtonItem:barButtonItem animate
    self.masterPopoverController = popoverController;
}

- (void)splitViewController:(UISplitViewController *)splitControlle
  willShowViewController:(UIViewController *)viewController
  invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    // Called when the view is shown again in the split view,
    //invalidating the button and popover controller.
    [self.navigationItem setLeftBarButtonItem:nil animated:YES];
    self.masterPopoverController = nil;
}

@end
```

10.现在当我们运行应用程序时，在横向模式下我们会得到下面的输出

11. 当我们切换到纵向模式，我们会获得下面的输出:

IOS文本视图的使用

文本视图用于显示多行滚动的文本。

重要属性

- dataDetectorTypes
- delegate
- editable
- inputAccessoryView
- inputView
- text
- textAlignment
- textColor

重要的委托方法

```
-(void)textViewDidBeginEditing:(UITextView *)textView
```

```
-(void)textViewDidEndEditing:(UITextView *)textView
```

```
-(void)textViewDidChange:(UITextView *)textView
```

```
-(BOOL)textViewShouldEndEditing:(UITextView *)textView
```

添加自定义方法 **addTextView**

```
-(void)addTextView{
    myTextView = [[UITextView alloc]initWithFrame:
    CGRectMake(10, 50, 300, 200)];
    [myTextView setText:@"Lorem ipsum dolor sit er elit lamet, cons
    cillium adipisicing pecu, sed do eiusmod tempor incididunt ut l
    dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exer
    ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis a
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    nulla pariatur. Excepteur sint occaecat cupidatat non proident,
    culpa qui officia deserunt mollit anim id est laborum. Nam libe
    conscient to factor tum poen legum odioque civiuda.
    Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipi
    pecu, sed do eiusmod tempor incididunt ut labore et dolore magn
    Ut enim ad minim veniam, quis nostrud exercitation ullamco labo
    aliquip ex ea commodo consequat. Duis aute irure dolor in repre
    in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum. Nam liber te c
    to factor tum poen legum odioque civiuda."];
    myTextView.delegate = self;
    [self.view addSubview:myTextView];

}
```

在 ViewController.m 中执行 textView 委托

```
#pragma mark - Text View delegates

-(BOOL)textView:(UITextView *)textView shouldChangeTextInRange:
 (NSRange)range replacementText:(NSString *)text{
    if ([text isEqualToString:@"\n"]) {
        [textView resignFirstResponder];
    }
    return YES;
}
-(void)textViewDidBeginEditing:(UITextView *)textView{
    NSLog(@"Did begin editing");
}
-(void)textViewDidChange:(UITextView *)textView{
    NSLog(@"Did Change");

}
-(void)textViewDidEndEditing:(UITextView *)textView{
    NSLog(@"Did End editing");

}
-(BOOL)textViewShouldEndEditing:(UITextView *)textView{
    [textView resignFirstResponder];
    return YES;
}
```

修改 ViewController.m 中的 viewDidLoad方法，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addTextView];
}
```

结果输出

现在当我们运行该应用程序我们会看到下面的输出

IOS视图切换的使用

视图切换通过一系列动画效果实现,包括折叠切换、爆炸切换、卡片式切换等等。

修改 ViewController.xib，如下所示

在 xib 中创建按钮的操作

修改 ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    UIView *view1;
    UIView *view2;
}

-(IBAction)flipFromLeft:(id)sender;
-(IBAction)flipFromRight:(id)sender;
-(IBAction)flipFromTop:(id)sender;
-(IBAction)flipFromBottom:(id)sender;
-(IBAction)curlUp:(id)sender;
-(IBAction)curlDown:(id)sender;
-(IBAction)dissolve:(id)sender;
-(IBAction)noTransition:(id)sender;

@end
```

在 ViewController 类中声明两个视图的实例。ViewController.h文件代码如下：

修改 ViewController.m

我们将添加自定义方法setUpView来初始化视图。

我们还将创建了另一种方法doTransitionWithType: 实现view1切换到view2，反之亦然。

后我们将执行之前创建的操作的方法即调用 doTransitionWithType: 方法与切换类型。ViewController.m代码如下:

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self setUpView];
    // Do any additional setup after loading the view, typically fr
}
```

```
-(void)setUpView{
    view1 = [[UIView alloc]initWithFrame:self.view.frame];
    view1.backgroundColor = [UIColor lightTextColor];
    view2 = [[UIView alloc]initWithFrame:self.view.frame];
    view2.backgroundColor = [UIColor orangeColor];
    [self.view addSubview:view1];
    [self.view sendSubviewToBack:view1];

}

-(void)doTransitionWithType:(UIViewAnimationTransition)animationTra
    if ([[self.view subviews] containsObject:view2 ]) {
        [UIView transitionFromView:view2
                            toView:view1
                          duration:2
                           options:animationTransitionType
                        completion:^(BOOL finished){
                            [view2 removeFromSuperview];
                        }];
        [self.view addSubview:view1];
        [self.view sendSubviewToBack:view1];
    }
    else{

        [UIView transitionFromView:view1
                            toView:view2
                          duration:2
                           options:animationTransitionType
                        completion:^(BOOL finished){
                            [view1 removeFromSuperview];
                        }];
        [self.view addSubview:view2];
        [self.view sendSubviewToBack:view2];

    }
}

-(IBAction)flipFromLeft:(id)sender
{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromRight:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromTop:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromBottom:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF
```

```
}
-(IBAction)curlUp:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCurlU

}
-(IBAction)curlDown:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCurlD

}
-(IBAction)dissolve:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCross

}
-(IBAction)noTransition:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionNone]

}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

输出

现在当我们运行该应用程序我们会看到下面的输出

您可以选择不同的按钮，看切换是如何工作。选择蜷缩切换将效果如下所示

选择器的使用

选择器是一个可滚动视图，用于选取列表项中的值。

重要的属性

- delegate
- dataSource

重要的方法

```
- (void)reloadAllComponents
```

```
- (void)reloadComponent:(NSInteger)component
```

```
- (NSInteger)selectedRowInComponent:(NSInteger)component
```

```
- (void)selectRow:(NSInteger)row inComponent:(NSInteger)component
  animated:(BOOL)animated
```

修改 ViewController.h

我们将添加一个文本字段、选择器视图和一个数组。

我们将采用UITextFieldDelegate、UIPickerViewDataSource、UIPickerViewDelegate的协议。ViewController.h文件代码如下所示：

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
<UITextFieldDelegate,UIPickerViewDataSource,UIPickerViewDelegate>
{

    UITextField *myTextField;
    UIPickerView *myPickerView;
    NSArray *pickerArray;
}
@end
```

添加自定义方法 addPickerView

```
-(void)addPickerView{
    pickerArray = [[NSArray alloc]initWithObjects:@"Chess",
    @"Cricket",@"Football",@"Tennis",@"Volleyball", nil];
    myTextField = [[UITextField alloc]initWithFrame:
    CGRectMake(10, 100, 300, 30)];
    myTextField.borderStyle = UITextBorderStyleRoundedRect;
    myTextField.textAlignment = UITextAlignmentCenter;
    myTextField.delegate = self;
    [self.view addSubview:myTextField];
    [myTextField setPlaceholder:@"Pick a Sport"];
    myPickerView = [[UIPickerView alloc]init];
    myPickerView.dataSource = self;
    myPickerView.delegate = self;
    myPickerView.showsSelectionIndicator = YES;
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc]
    initWithTitle:@"Done" style:UIBarButtonItemStyleDone
    target:self action:@selector(done:)];
    UIToolbar *toolBar = [[UIToolbar alloc]initWithFrame:
    CGRectMake(0, self.view.frame.size.height-
    myDatePicker.frame.size.height-50, 320, 50)];
    [toolBar setBarStyle:UIBarStyleBlackOpaque];
    NSArray *toolbarItems = [NSArray arrayWithObjects:
    doneButton, nil];
    [toolBar setItems:toolbarItems];
    myTextField.inputView = myPickerView;
    myTextField.inputAccessoryView = toolBar;

}
```

执行委托，如下所示:

```
#pragma mark - Text field delegates

-(void)textFieldDidBeginEditing:(UITextField *)textField{
    if ([textField.text isEqualToString:@""]) {
        [self dateChanged:nil];
    }
}
#pragma mark - Picker View Data source
-(NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerVi
    return 1;
}
-(NSInteger)pickerView:(UIPickerView *)pickerView
 numberOfRowsInComponent:(NSInteger)component{
    return [pickerArray count];
}

#pragma mark- Picker View Delegate

-(void)pickerView:(UIPickerView *)pickerView didSelectRow:
 (NSInteger)row inComponent:(NSInteger)component{
    [myTextField setText:[pickerArray objectAtIndex:row]];
}
- (NSString *)pickerView:(UIPickerView *)pickerView titleForRow:
  (NSInteger)row forComponent:(NSInteger)component{
    return [pickerArray objectAtIndex:row];
}
```

在ViewController.m修改viewDidLoad，如下所示:

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addPickerView];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出:

文本选择器视图如下所示，我们可以选取我们需要的值：

IOS开关的使用

开关用于打开和关闭状态之间的切换。

重要的属性

- onImage
- offImage
- on

重要的方法

```
- (void)setOn:(BOOL)on animated:(BOOL)animated
```

添加自定义方法 addSwitch 和开关

```
-(IBAction)switched:(id)sender{
    NSLog(@"Switch current state %@", mySwitch.on ? @"On" : @"Off")
}
-(void)addSwitch{
    mySwitch = [[UISwitch alloc] init];
    [self.view addSubview:mySwitch];
    mySwitch.center = CGPointMake(150, 200);
    [mySwitch addTarget:self action:@selector(switched:)
    forControlEvents:UIControlEventValueChanged];
}
```

在 ViewController.m 中修改 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addSwitch];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出

向右滑动开关输出如下所示

IOS滑块的使用

滑块用于从某个范围的值里选择的一个值。

重要的属性

- continuous
- maximumValue
- minimumValue
- value

重要的方法

```
- (void)setValue:(float)value animated:(BOOL)animated
```

添加自定义方法 **addSlider** 和 **sliderChanged**

```
-(IBAction)sliderChanged:(id)sender{
    NSLog(@"SliderValue %f",mySlider.value);
}
-(void)addSlider{
    mySlider = [[UISlider alloc] initWithFrame:CGRectMake(50, 200,
    [self.view addSubview:mySlider];
    mySlider.minimumValue = 10.0;
    mySlider.maximumValue = 99.0;
    mySlider.continuous = NO;
    [mySlider addTarget:self action:@selector(sliderChanged:)
    forControlEvents:UIControlEventValueChanged];
}
```

在 **ViewController.m** 中修改 **viewDidLoad**，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addSlider];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出

当拖动滑块效果如下：



IOS警告对话框的使用

警告对话框用来给用户提供重要信息。

仅在警告对话框视图中选择选项后，才能着手进一步使用应用程序。

重要的属性

- alertViewStyle
- cancelButtonIndex
- delegate
- message
- numberOfButtons
- title

重要的方法

```
- (NSInteger)addButtonWithTitle:(NSString *)title
```

```
- (NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex
```

```
- (void)dismissWithClickedButtonIndex:
  (NSInteger)buttonIndex animated:(BOOL)animated
```

```
- (id)initWithTitle:(NSString *)title message:
  (NSString *)message delegate:(id)delegate
  cancelButtonTitle:(NSString *)cancelButtonTitle
  otherButtonTitles:(NSString*)otherButtonTitles, ...
```

```
- (void)show
```

更新 **ViewController.h**，如下所示

让类符合警告对话框视图的委托协议，如下所示，在ViewController.h中添加 <UIAlertViewDelegate>

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIAlertViewDelegate>{

}
@end
```

添加自定义方法 addAlertView

```
-(void)addAlertView{
    UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:
    @"Title" message:@"This is a test alert" delegate:self
    cancelButtonTitle:@"Cancel" otherButtonTitles:@"Ok", nil];
    [alertView show];
}
```

执行警告对话框视图的委托方法

```
#pragma mark - Alert view delegate
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
  (NSInteger)buttonIndex{
    switch (buttonIndex) {
        case 0:
            NSLog(@"Cancel button clicked");
            break;
        case 1:
            NSLog(@"OK button clicked");
            break;

        default:
            break;
    }
}
```

在 ViewController.m 中修改 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addAlertView];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出：



IOS图标的使用

IOS图标是用于应用程序相关的操作。

IOS 中的不同图标

- AppIcon
- App Store 的应用程序图标
- 搜索结果和设置的小图标
- 工具栏和导航栏图标
- 选项卡栏图标

AppIcon

AppIcon 是出现在设备SpringBoard （默认屏幕上的所有的应用程序） 的应用程序的图标。

App Store 的应用程序图标

它是512 x 512 或 1024 x 1024(推荐大小)的高分辨率的应用程序图标。

搜索结果和设置的小图标

在搜索列表的应用程序中使用这个小图标。

它还用于与相关的应用程序的功能是启用和禁用的设置屏幕上。如：启用定位服务。

工具栏和导航栏图标

工具栏和导航栏中使用特制的标准图标的列表。它包括的份额，像图标相机，撰写等等。

选项卡栏图标

选项卡栏中使用一系列特制的标准图标列表。它包括的图标有书签、 联系人、 下载等。

有的不同的 iOS 设备的每个图标大小的都不一样。你可以查看更多关于苹果文件中图标的准则：ios人机交互界面指南。

IOS加速度传感器(accelerometer)

简介

加速度传感器是根据x、y和z三个方向来检测在设备位置的改变。

通过加速度传感器可以知道当前设备相对于地面的位置。

以下实例代码需要在真实设备上运行，在模拟器上是无法工作的。

实例步骤

1、创建一个简单的视图应用程序

2、在ViewController.xib中添加三个标签，并创建一个ibOutlets分别为：xlable、ylabel和zlabel

3、如下所示，更新ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIAccelerometerDelegat
{
    IBOutlet UILabel *xlabel;
    IBOutlet UILabel *ylabel;
    IBOutlet UILabel *zlabel;

}
@end
```

4、如下所示，更新ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [[UIAccelerometer sharedAccelerometer]setDelegate:self];
    //Do any additional setup after loading the view,typically fror
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerat
  (UIAcceleration *)acceleration{
    [xlabel setText:[NSString stringWithFormat:@"%f",acceleration.x
    [ylabel setText:[NSString stringWithFormat:@"%f",acceleration.y
    [zlabel setText:[NSString stringWithFormat:@"%f",acceleration.z
}

@end
```

输出

当我们在iPhone设备中运行该应用程序，得到的输出结果如下所示。


No SIM
9:01 AM
87%
-0.036697
-0.886536
-0.421844

IOS通用应用程序

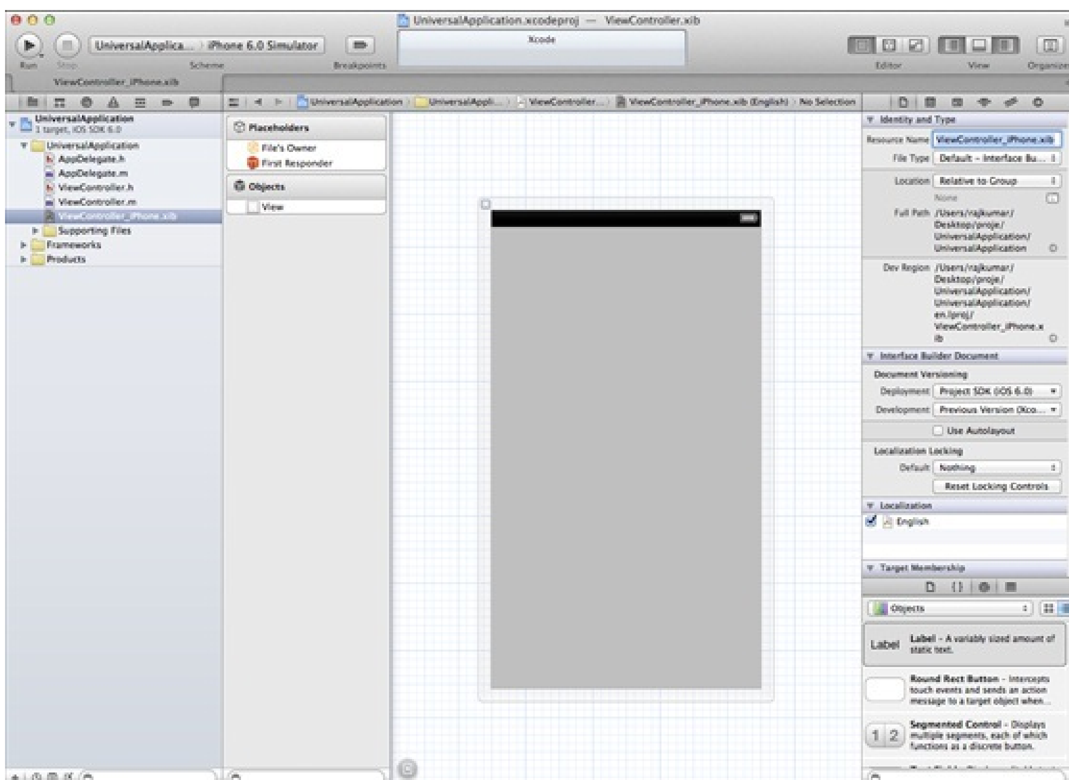
简介

通用的应用程序是为iPhone和iPad在一个单一的二进制文件中设计的应用程序。这有助于代码重用，并能够帮助更快进行更新。

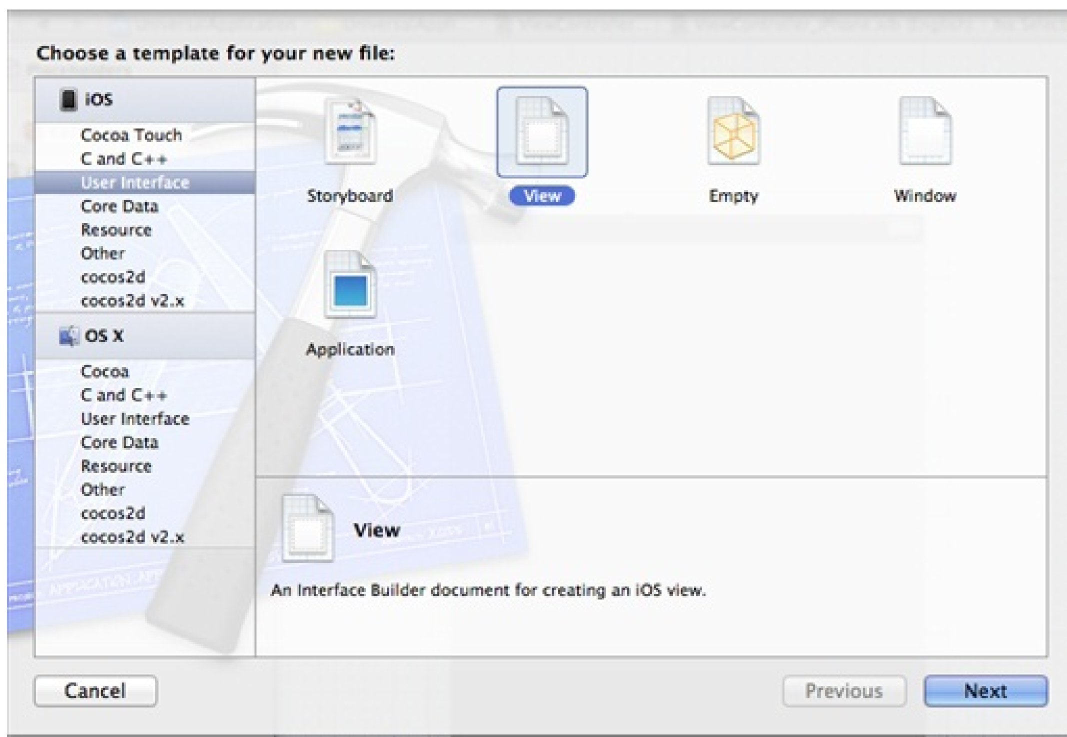
实例步骤

1、创建一个简单的View based application（视图应用程序）

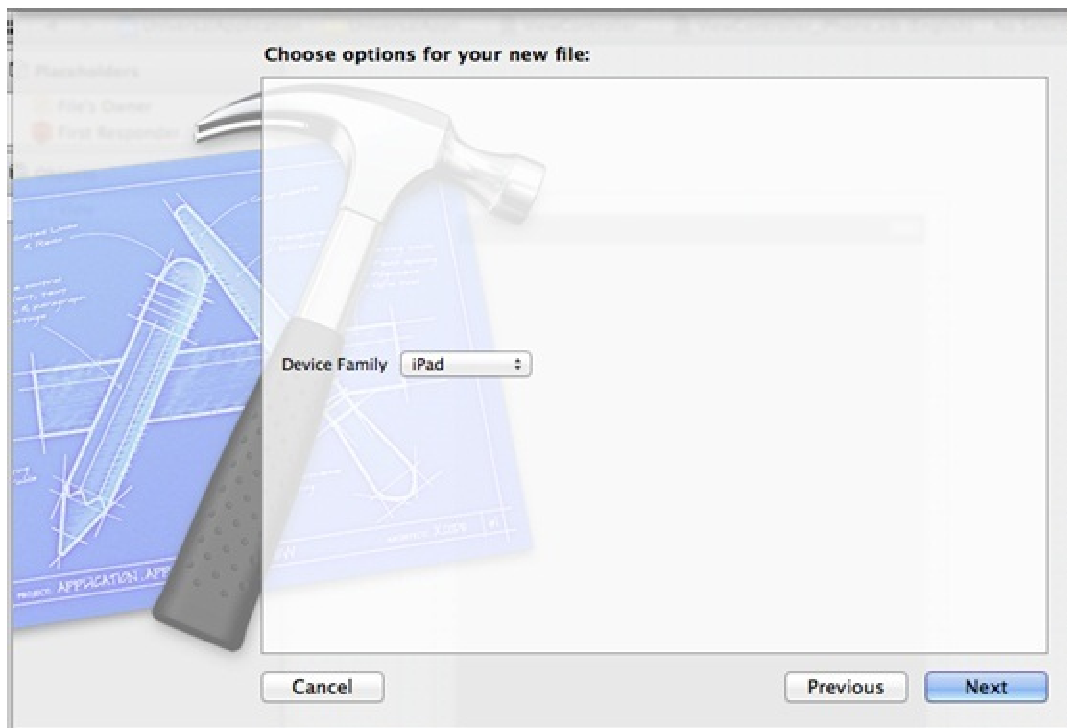
2、在文件查看器的右边，将文件ViewController.xib的文件名称更改为ViewController_iPhone.xib，如下所示



3、选择"File -> New -> File... "，然后选择User Interface，再选择View，单击下一步



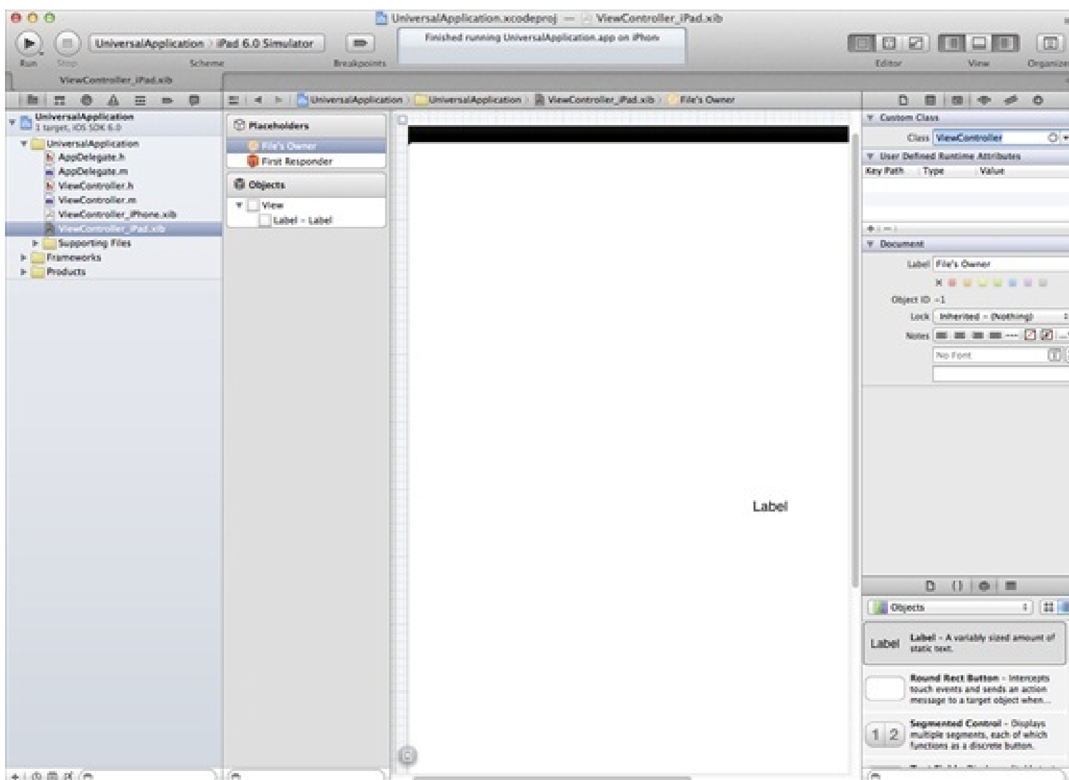
4、选择iPad作为设备，单击下一步:



5、将该文件另存为ViewController_iPad.xib，然后选择创建

6、在ViewController_iPhone.xib和ViewController_iPad.xibd的屏幕中心添加标签

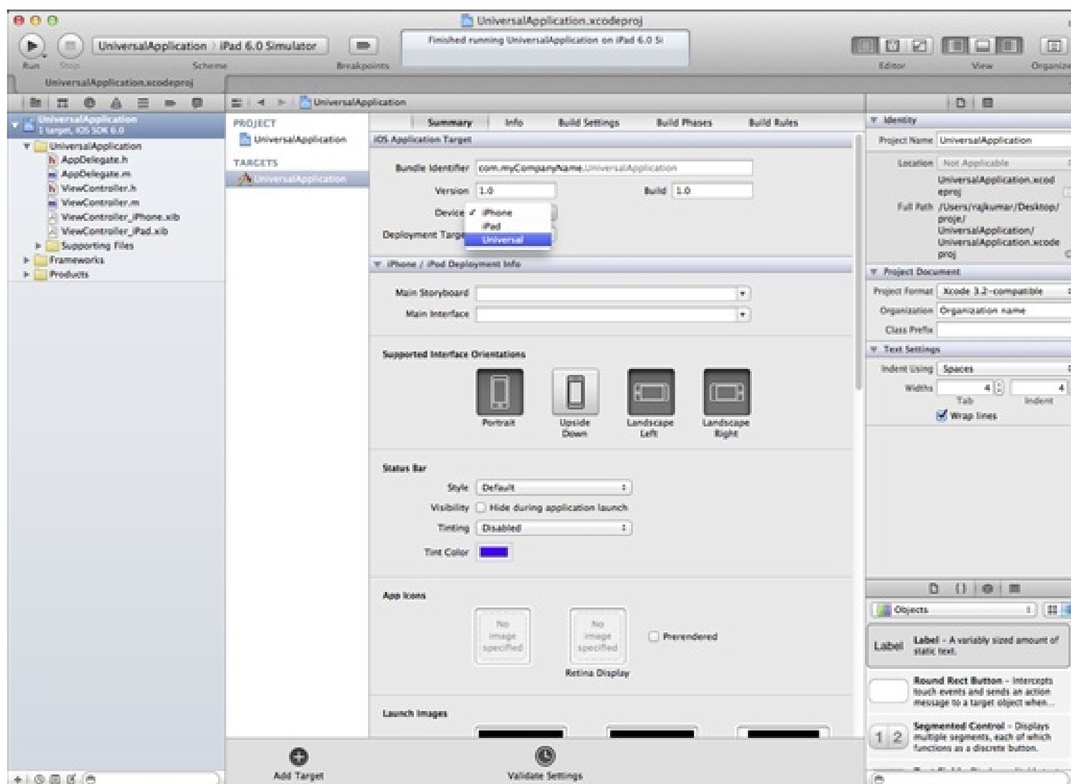
7、在ViewController_iPhone.xib中选择identity inspector，设置custom class为ViewController



8、更新AppDelegate.m中的 application:DidFinishLaunching:withOptions方法

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
   self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
   mainScreen] bounds]];
   // Override point for customization after application launch.
   if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPhone) {
        self.viewController = [[ViewController alloc]
        initWithNibName:@"ViewController_iPhone" bundle:nil];
   }
   else{
        self.viewController = [[ViewController alloc] initWithNibNa
        @"ViewController_iPad" bundle:nil];
   }
   self.window.rootViewController = self.viewController;
   [self.window makeKeyAndVisible];
   return YES;
}
```

9、在项目摘要中更新设备中为universal，如下所示：

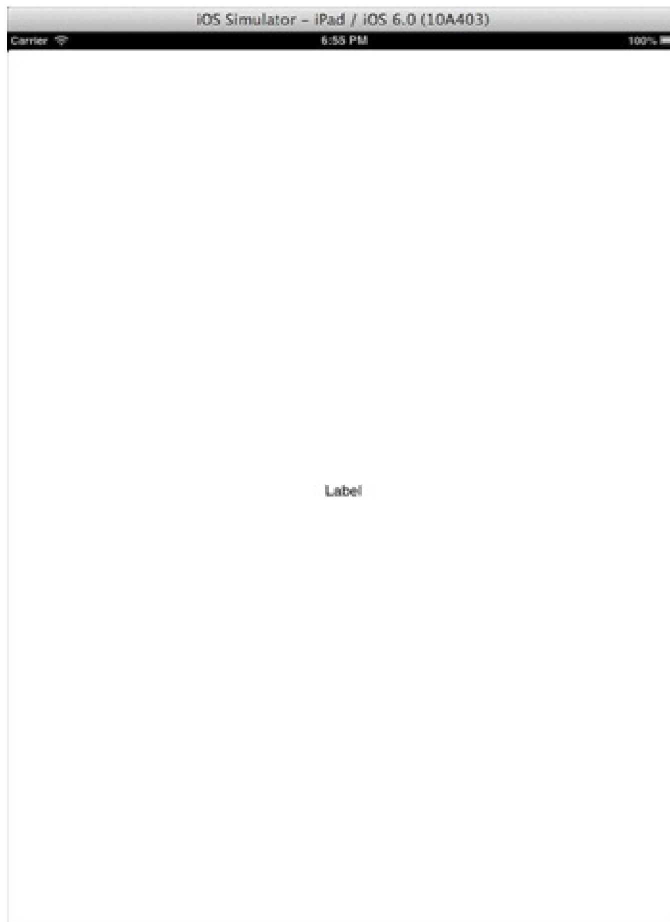


输出

运行该应用程序，我们会看到下面的输出



在iPad模拟器中运行应用程序,我们会得到下面的输出:



IOS相机管理

相机简介

相机是移动设备的共同特点之一，我们能够使用相机拍摄图片，并在应用程序里调用它，而且相机的使用很简单。

实例步骤

1、创建一个简单的View based application

2、在ViewController.xib中添加一个button （按钮），并为该按钮创建IBAction

3、添加一个 image view （图像视图），并创建一个名为imageView的IBOutlet

4、ViewController.h文件代码如下所示：

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIImagePickerControlle
{
   UIImagePickerController *imagePicker;
   IBOutlet UIImageView *imageView;
}
- (IBAction)showCamera:(id)sender;

@end
```

5、修改ViewController.m,如下所示：

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)showCamera:(id)sender {
    imagePicker.allowsEditing = YES;
    if ([UIImagePickerController isSourceTypeAvailable:
        UIImagePickerControllerSourceTypeCamera])
    {
        imagePicker.sourceType = UIImagePickerControllerSourceType(
    }
    else{
        imagePicker.sourceType =
        UIImagePickerControllerSourceTypePhotoLibrary;
    }
    [self presentModalViewController:imagePicker animated:YES];

}
-(void)imagePickerController:(UIImagePickerController *)picker
  didFinishPickingMediaWithInfo:(NSDictionary *)info{
    UIImage *image = [info objectForKey:UIImagePickerControllerEdit
    if (image == nil) {
        image = [info objectForKey:UIImagePickerControllerOriginalI
    }
    imageView.image = image;

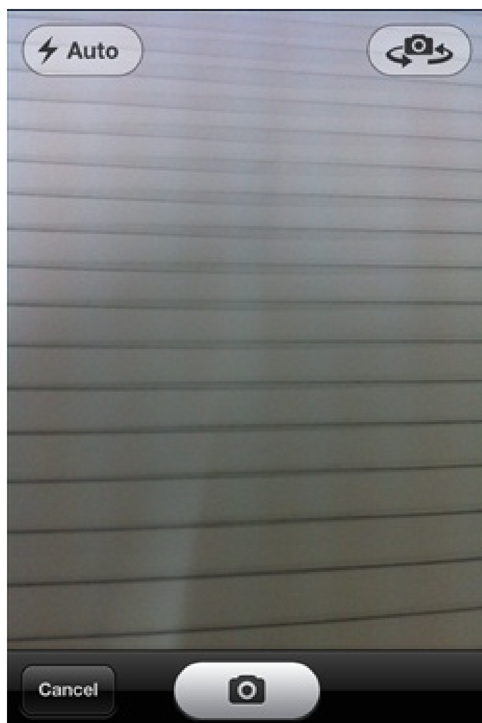
}

-(void)imagePickerControllerDidCancel:(UIImagePickerController *)pi
    [self dismissModalViewControllerAnimated:YES];
}

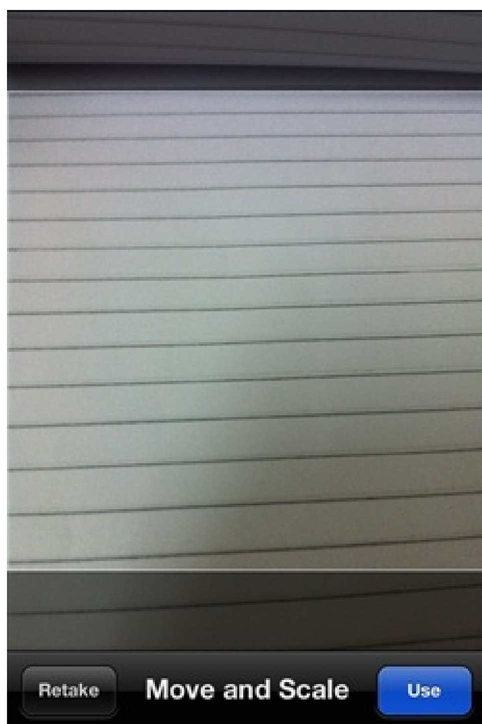
@end
```

输出

运行该应用程序并单击显示相机按钮时，我们就会获得下面的输出



只要拍照之后，就可以通过移动和缩放对图片进行编辑，如下所示。



IOS定位操作

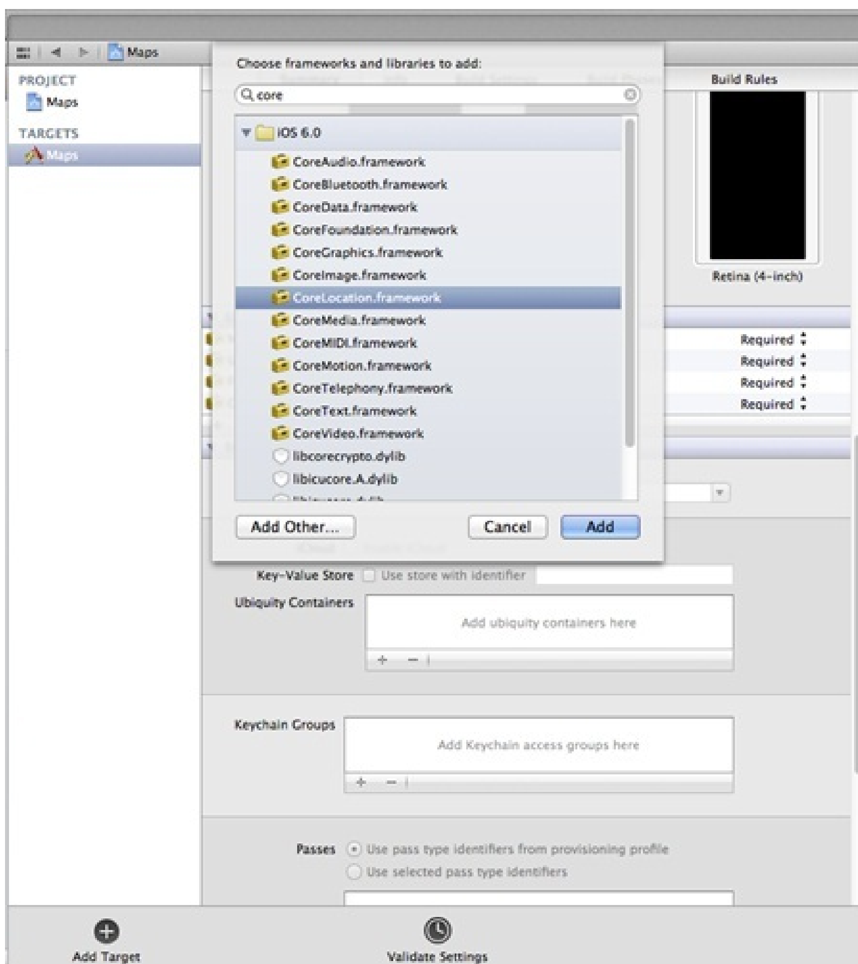
简介

在IOS中通过CoreLocation定位，可以获取到用户当前位置，同时能得到装置移动信息。

实例步骤

1、创建一个简单的View based application（视图应用程序）。

2、择项目文件，然后选择目标，然后添加CoreLocation.framework,如下所示



3、在ViewController.xib中添加两个标签，创建ibOutlet名为latitudeLabel和longtitudeLabel的标签

4、现在通过选择" File-> New -> File... -> "选择Objective C class 并单击下一步

5、把"sub class of"作为NSObject，将类命名为LocationHandler

6、选择创建

7、更新LocationHandler.h，如下所示

```
#import <Foundation/Foundation.h>
#import <CoreLocation/CoreLocation.h>

@protocol LocationHandlerDelegate <NSObject>

@required
-(void) didUpdateToLocation:(CLLocation*)newLocation
 fromLocation:(CLLocation*)oldLocation;
@end

@interface LocationHandler : NSObject<CLLocationManagerDelegate>
{
    CLLocationManager *locationManager;
}
@property(nonatomic,strong) id<LocationHandlerDelegate> delegate;

+(id)getSharedInstance;
-(void)startUpdating;
-(void) stopUpdating;

@end
```

8、更新LocationHandler.m,如下所示

```
#import "LocationHandler.h"
static LocationHandler *DefaultManager = nil;

@interface LocationHandler()

-(void)initiate;

@end

@implementation LocationHandler

+(id)getSharedInstance{
    if (!DefaultManager) {
        DefaultManager = [[self allocWithZone:NULL]init];
        [DefaultManager initiate];
    }
    return DefaultManager;
}
-(void)initiate{
    locationManager = [[CLLocationManager alloc]init];
    locationManager.delegate = self;
}

-(void)startUpdating{
    [locationManager startUpdatingLocation];
}

-(void) stopUpdating{
    [locationManager stopUpdatingLocation];
}
-(void)locationManager:(CLLocationManager *)manager didUpdateToLoca
 (CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation{
    if ([self.delegate respondsToSelector:@selector
    (didUpdateToLocation:fromLocation:)])
    {
        [self.delegate didUpdateToLocation:oldLocation
        fromLocation:newLocation];

    }
}

@end
```

9、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import "LocationHandler.h"
@interface ViewController : UIViewController<LocationHandlerDelegat
{
    IBOutlet UILabel *latitudeLabel;
    IBOutlet UILabel *longitudeLabel;
}
@end
```

10、更新ViewController.m,如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [[LocationHandler getSharedInstance]setDelegate:self];
    [[LocationHandler getSharedInstance]startUpdating];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

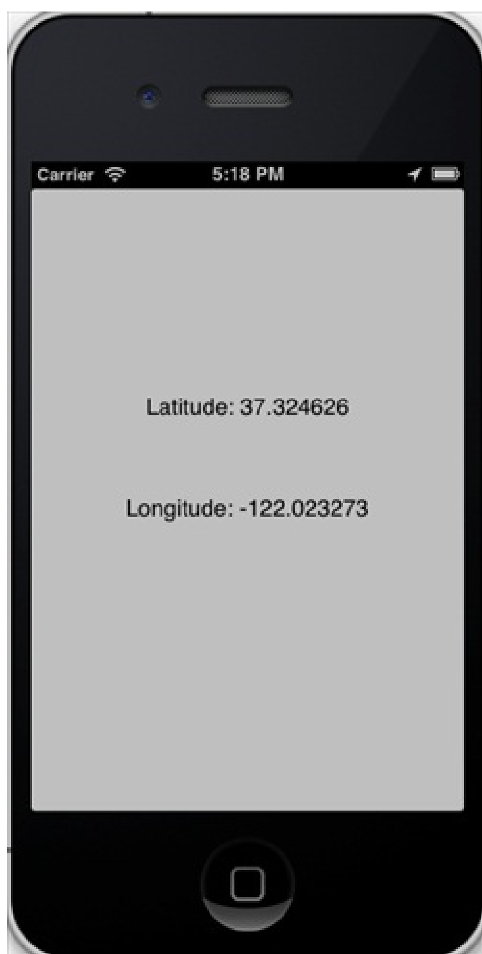
-(void)didUpdateToLocation:(CLLocation *)newLocation
 fromLocation:(CLLocation *)oldLocation{
    [latitudeLabel setText:[NSString stringWithFormat:
    @"Latitude: %f",newLocation.coordinate.latitude]];
    [longitudeLabel setText:[NSString stringWithFormat:
    @"Longitude: %f",newLocation.coordinate.longitude]];

}

@end
```

输出

当我们运行该应用程序，会得到如下的输出:


Carrier
5:18 PM
Latitude: 37.324626
Longitude: -122.023273

IOS SQLite数据库

简介

在IOS中使用Sqlite来处理数据。如果你已经了解了SQL，那你可以很容易的掌握SQLite数据库的操作。

实例步骤

1、创建一个简单的View based application

2、选择项目文件，然后选择目标，添加libsqlite3.dylib库到选择框架

3、通过选择" File-> New -> File... -> "选择 Objective C class 创建新文件，单击下一步

4、"sub class of"为NSObject"，类命名为DBManager

5、选择创建

6、更新DBManager.h,如下所示

```
#import <Foundation/Foundation.h>
#import <sqlite3.h>

@interface DBManager : NSObject
{
    NSString *databasePath;
}

+(DBManager*)getSharedInstance;
-(BOOL)createDB;
-(BOOL) saveData:(NSString*)registerNumber name:(NSString*)name
  department:(NSString*)department year:(NSString*)year;
-(NSArray*) findByRegisterNumber:(NSString*)registerNumber;

@end
```

7、更新DBManager.m,如下所示

```
#import "DBManager.h"
static DBManager *sharedInstance = nil;
static sqlite3 *database = nil;
static sqlite3_stmt *statement = nil;

@implementation DBManager
```

```
+(DBManager*)getSharedInstance{
    if (!sharedInstance) {
        sharedInstance = [[super allocWithZone:NULL]init];
        [sharedInstance createDB];
    }
    return sharedInstance;
}

-(BOOL)createDB{
    NSString *docsDir;
    NSArray *dirPaths;
    // Get the documents directory
    dirPaths = NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES);
    docsDir = dirPaths[0];
    // Build the path to the database file
    databasePath = [[NSString alloc] initWithString:
    [docsDir stringByAppendingPathComponent: @"student.db"]];
    BOOL isSuccess = YES;
    NSFileManager *filemgr = [NSFileManager defaultManager];
    if ([filemgr fileExistsAtPath: databasePath ] == NO)
    {
        const char *dbpath = [databasePath UTF8String];
        if (sqlite3_open(dbpath, &database) == SQLITE_OK)
        {
            char *errMsg;
            const char *sql_stmt =
            "create table if not exists studentsDetail (regno integ
            primary key, name text, department text, year text)";
            if (sqlite3_exec(database, sql_stmt, NULL, NULL, &errMs
               != SQLITE_OK)
            {
                isSuccess = NO;
                NSLog(@"Failed to create table");
            }
            sqlite3_close(database);
            return  isSuccess;
        }
        else {
            isSuccess = NO;
            NSLog(@"Failed to open/create database");
        }
    }
    return isSuccess;
}

- (BOOL) saveData:(NSString*)registerNumber name:(NSString*)name
  department:(NSString*)department year:(NSString*)year;
{
    const char *dbpath = [databasePath UTF8String];
    if (sqlite3_open(dbpath, &database) == SQLITE_OK)
    {
```

```
        NSString *insertSQL = [NSString stringWithFormat:@"insert i
        studentsDetail (regno,name, department, year) values
        (\"%d\",\"%@\", \"%@\", \"%@\")",[registerNumber integerVal
        name, department, year];
        const char *insert_stmt = [insertSQL UTF8String];
        sqlite3_prepare_v2(database, insert_stmt,-1, &statement, NU
        if (sqlite3_step(statement) == SQLITE_DONE)
        {
            return YES;
        }
        else {
            return NO;
        }
        sqlite3_reset(statement);
    }
    return NO;
}

- (NSArray*) findByRegisterNumber:(NSString*)registerNumber
{
    const char *dbpath = [databasePath UTF8String];
    if (sqlite3_open(dbpath, &database) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat:
        @"select name, department, year from studentsDetail where
        regno=\"%@\"",registerNumber];
        const char *query_stmt = [querySQL UTF8String];
        NSMutableArray *resultArray = [[NSMutableArray alloc]init];
        if (sqlite3_prepare_v2(database,
           query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            if (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSString *name = [[NSString alloc] initWithUTF8Stri
                 (const char *) sqlite3_column_text(statement, 0)];
                [resultArray addObject:name];
                NSString *department = [[NSString alloc] initWithUT
                (const char *) sqlite3_column_text(statement, 1)];
                [resultArray addObject:department];
                NSString *year = [[NSString alloc]initWithUTF8Strin
                (const char *) sqlite3_column_text(statement, 2)];
                [resultArray addObject:year];
                return resultArray;
            }
            else{
                NSLog(@"Not found");
                return nil;
            }
            sqlite3_reset(statement);
        }
    }
    return nil;
}
```



8、如图所示，更新ViewController.xib文件

9、为上述文本字段创建IBOutlets

10、为上述按钮创建IBAction

11、如下所示，更新ViewController.h

```
#import <UIKit/UIKit.h>
#import "DBManager.h"

@interface ViewController : UIViewController<UITextFieldDelegate>
{
    IBOutlet UITextField *regNoTextField;
    IBOutlet UITextField *nameTextField;
    IBOutlet UITextField *departmentTextField;
    IBOutlet UITextField *yearTextField;
    IBOutlet UITextField *findByRegisterNumberTextField;
    IBOutlet UIScrollView *myScrollView;
}

-(IBAction)saveData:(id)sender;
-(IBAction)findData:(id)sender;

@end
```

12、更新ViewController.m,如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
  nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNi
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
-(IBAction)saveData:(id)sender{
    BOOL success = NO;
    NSString *alertString = @"Data Insertion failed";
    if (regNoTextField.text.length>0 &&nameTextField.text.length>0
    departmentTextField.text.length>0 &&yearTextField.text.length>0
    {
        success = [[DBManager getSharedInstance]saveData:
        regNoTextField.text name:nameTextField.text department:
        departmentTextField.text year:yearTextField.text];
    }
    else{
        alertString = @"Enter all fields";
    }
    if (success == NO) {
        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:
        alertString message:nil
        delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil]
        [alert show];
    }
}

-(IBAction)findData:(id)sender{
    NSArray *data = [[DBManager getSharedInstance]findByRegisterNum
    findByRegisterNumberTextField.text];
    if (data == nil) {
        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:
        @"Data not found" message:nil delegate:nil cancelButtonTitl
        @"OK" otherButtonTitles:nil];
        [alert show];
        regNoTextField.text = @"";
        nameTextField.text =@"";
        departmentTextField.text = @"";
        yearTextField.text =@"";
    }
    else{
        regNoTextField.text = findByRegisterNumberTextField.text;
        nameTextField.text =[data objectAtIndex:0];
        departmentTextField.text = [data objectAtIndex:1];
        yearTextField.text =[data objectAtIndex:2];
    }
}

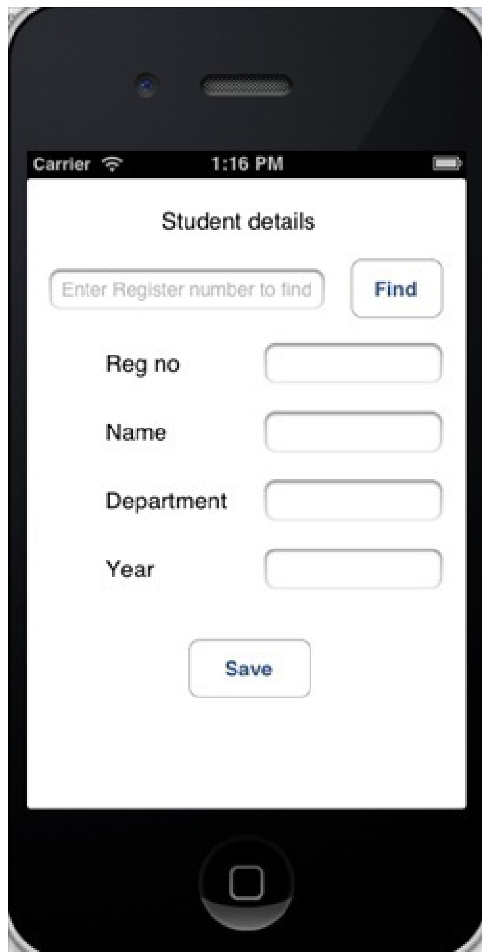
#pragma mark - Text field delegate
-(void)textFieldDidBeginEditing:(UITextField *)textField{
    [myScrollView setFrame:CGRectMake(10, 50, 300, 200)];
    [myScrollView setContentSize:CGSizeMake(300, 350)];
}
-(void)textFieldDidEndEditing:(UITextField *)textField{
    [myScrollView setFrame:CGRectMake(10, 50, 300, 350)];

}
-(BOOL) textFieldShouldReturn:(UITextField *)textField{
```

```
    [textField resignFirstResponder];
    return YES;
}
@end
```

输出

现在当我们运行应用程序时，我们就会获得下面的输出，我们可以在其中添加及查找学生的详细信息



IOS发送电子邮件

简介

我们可以使用IOS设备中的电子邮件应用程序发送电子邮件。

实例步骤

1、创建一个简单的View based application

2、选择项目文件，然后选择目标，然后添加MessageUI.framework

3、在ViewController.xib中添加一个按钮，创建用于发送电子邮件的操作（action）

4、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>

@interface ViewController : UIViewController<MFMailComposeViewContr
{
    MFMailComposeViewController *mailComposer;
}

-(IBAction)sendMail:(id)sender;

@end
```

5、如下所示，更新ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(void)sendMail:(id)sender{
    mailComposer = [[MFMailComposeViewController alloc]init];
    mailComposer.mailComposeDelegate = self;
    [mailComposer setSubject:@"Test mail"];
    [mailComposer setMessageBody:@"Testing message
    for the test mail" isHTML:NO];
    [self presentModalViewController:mailComposer animated:YES];
}

#pragma mark - mail compose delegate
-(void)mailComposeController:(MFMailComposeViewController *)control
 didFinishWithResult:(MFMailComposeResult)result error:(NSError *)e
   if (result) {
        NSLog(@"Result : %d",result);
    }
    if (error) {
        NSLog(@"Error : %@",error);
    }
    [self dismissModalViewControllerAnimated:YES];

}

@end
```

输出

当运行该应用程序，会看如下的输出结果



当点击"send email"发送按钮后，可以看到如下结果：


Carrier
4:41 PM
Cancel
Test mail
Send
To:
Cc/Bcc:
Subject: Test mail
Testing message for the test mail
Sent from my iPhone Simulator

IOS音频和视频(Audio & Video)

简介

音频和视频在最新的设备中颇为常见。

将iosAVFoundation.framework和MediaPlayer.framework添加到Xcode项目中，可以让IOS支持音频和视频(Audio & Video)。

实例步骤

1、创建一个简单的View based application

2、选择项目文件、选择目标，然后添加AVFoundation.framework和MediaPlayer.framework

3、在ViewController.xib中添加两个按钮，创建一个用于分别播放音频和视频的动作（action）

4、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <MediaPlayer/MediaPlayer.h>

@interface ViewController : UIViewController
{
    AVAudioPlayer *audioPlayer;
    MPMoviePlayerViewController *moviePlayer;

}
-(IBAction)playAudio:(id)sender;
-(IBAction)playVideo:(id)sender;
@end
```

5、更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

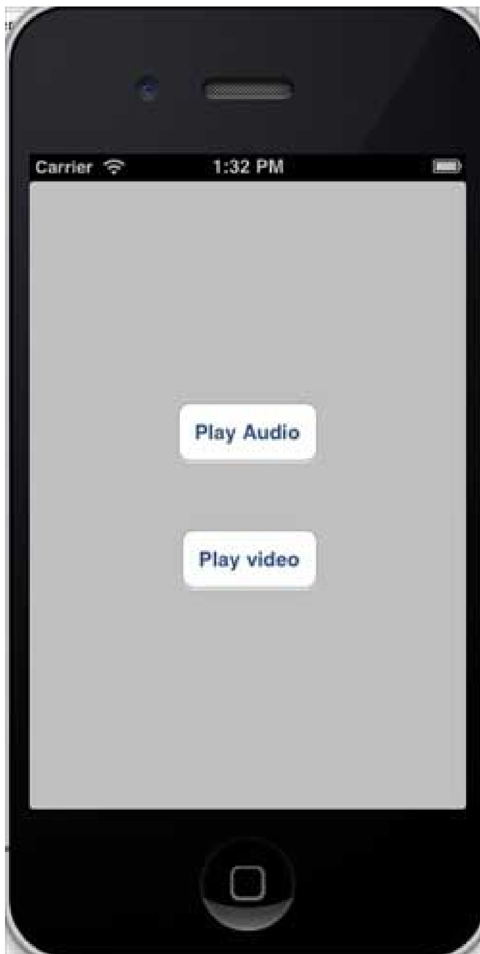
- (void)didReceiveMemoryWarning
{
   [super didReceiveMemoryWarning];
   // Dispose of any resources that can be recreated.
}
-(IBAction)playAudio:(id)sender{
   NSString *path = [[NSBundle mainBundle]
   pathForResource:@"audioTest" ofType:@"mp3"];
   audioPlayer = [[AVAudioPlayer alloc]initWithContentsOfURL:
   [NSURL fileURLWithPath:path] error:NULL];
   [audioPlayer play];
}
-(IBAction)playVideo:(id)sender{
   NSString *path = [[NSBundle mainBundle]pathForResource:
   @"videoTest" ofType:@"mov"];
   moviePlayer = [[MPMoviePlayerViewController
   alloc]initWithContentURL:[NSURL fileURLWithPath:path]];
   [self presentModalViewController:moviePlayer animated:NO];
}
@end
```

注意项

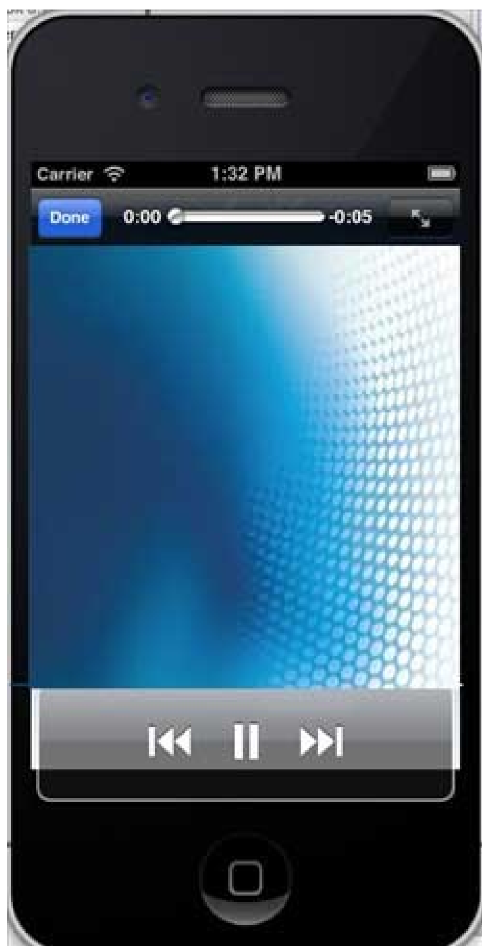
需要添加音频和视频文件，以确保获得预期的输出

输出

运行该程序，得到的输出结果如下



当我们点击 play video(播放视频)显示如下：


Carrier
1:32 PM
Done
0:00
-0:05

IOS文件处理

简介

文件处理不能直观的通过应用程序来解释，我们可以从以下实例来了解IOS的文件处理。

IOS中对文件的操作. 因为应用是在沙箱（sandbox）中的，在文件读写权限上受到限制，只能在几个目录下读写文件。

文件处理中使用的方法

下面列出了用于访问和操作文件的方法的列表。

以下实例你必须替换FilePath1、FilePath和FilePath字符串为完整的文件路径，以获得所需的操作。

检查文件是否存在

```
NSFileManager *fileManager = [NSFileManager defaultManager];
//Get documents directory
NSArray *directoryPaths = NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectoryPath = [directoryPaths objectAtIndex
if ([fileManager fileExistsAtPath:@""]==YES) {
     NSLog(@"File exists");
 }
```

比较两个文件的内容

```
if ([fileManager contentsEqualAtPath:@"FilePath1" andPath:@" Fil
   NSLog(@"Same content");
}
```

检查是否可写、可读、可执行文件

```
if ([fileManager isWritableFileAtPath:@"FilePath"]) {
      NSLog(@"isWritable");
  }
  if ([fileManager isReadableFileAtPath:@"FilePath"]) {
      NSLog(@"isReadable");
  }
  if ( [fileManager isExecutableFileAtPath:@"FilePath"]){
      NSLog(@"is Executable");
  }
```

移动文件

```
  if([fileManager moveItemAtPath:@"FilePath1"
  toPath:@"FilePath2" error:NULL]){
      NSLog(@"Moved successfully");
  }
```

复制文件

```
  if ([fileManager copyItemAtPath:@"FilePath1"
  toPath:@"FilePath2"  error:NULL]) {
      NSLog(@"Copied successfully");
  }
```

删除文件

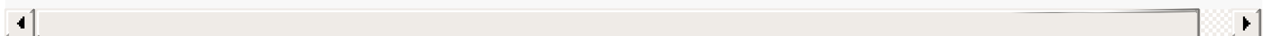
```
if ([fileManager removeItemAtPath:@"FilePath" error:NULL]) {
      NSLog(@"Removed successfully");
  }
```

读取文件

```
NSData *data = [fileManager contentsAtPath:@"Path"];
```

写入文件

```
 [fileManager createFileAtPath:@"" contents:data attributes:nil];
```



IOS地图开发

简介

IOS地图帮助我们定位位置，IOS地图使用 MapKit 框架。

实例步骤

1.创建一个简单的 View based application

2.选择项目文件，然后选择目标，然后添加MapKit.framework.

3.添加 Corelocation.framework

4.向 ViewController.xib 添加地图查看和创建 ibOutlet 并且命名为mapView。

5.通过"File-> New -> File... -> "选择 Objective C class创建一个新的文件，单击下一步

6."sub class of"为 NSObject，类作命名为MapAnnotation

7.选择创建

8.更新MapAnnotation.h，如下所示

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface MapAnnotation : NSObject<MKAnnotation>
@property (nonatomic, strong) NSString *title;
@property (nonatomic, readwrite) CLLocationCoordinate2D coordinate;

- (id)initWithTitle:(NSString *)title andCoordinate:
  (CLLocationCoordinate2D)coordinate2d;

@end
```

9.更新MapAnnotation.m，如下所示

```
#import "MapAnnotation.h"

@implementation MapAnnotation
-(id)initWithTitle:(NSString *)title andCoordinate:
 (CLLocationCoordinate2D)coordinate2d{
    self.title = title;
    self.coordinate =coordinate2d;
    return self;
}
@end
```

10.更新ViewController.h， 如下所示

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>
#import <CoreLocation/CoreLocation.h>
@interface ViewController : UIViewController<MKMapViewDelegate>
{
    MKMapView *mapView;
}
@end
```

11.更新ViewController.m， 如下所示

```
#import "ViewController.h"
#import "MapAnnotation.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
   [super viewDidLoad];
   mapView = [[MKMapView alloc]initWithFrame:
   CGRectMake(10, 100, 300, 300)];
   mapView.delegate = self;
   mapView.centerCoordinate = CLLocationCoordinate2DMake(37.32, -12
   mapView.mapType = MKMapTypeHybrid;
   CLLocationCoordinate2D location;
   location.latitude = (double) 37.332768;
   location.longitude = (double) -122.030039;
   // Add the annotation to our map view
   MapAnnotation *newAnnotation = [[MapAnnotation alloc]
   initWithTitle:@"Apple Head quaters" andCoordinate:location];
   [mapView addAnnotation:newAnnotation];
   CLLocationCoordinate2D location2;
   location2.latitude = (double) 37.35239;
   location2.longitude = (double) -122.025919;
   MapAnnotation *newAnnotation2 = [[MapAnnotation alloc]
   initWithTitle:@"Test annotation" andCoordinate:location2];
   [mapView addAnnotation:newAnnotation2];
   [self.view addSubview:mapView];
}
// When a map annotation point is added, zoom to it (1500 range)
- (void)mapView:(MKMapView *)mv didAddAnnotationViews:(NSArray *)vi
{
   MKAnnotationView *annotationView = [views objectAtIndex:0];
   id <MKAnnotation> mp = [annotationView annotation];
   MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance
   ([mp coordinate], 1500, 1500);
   [mv setRegion:region animated:YES];
   [mv selectAnnotation:mp animated:YES];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

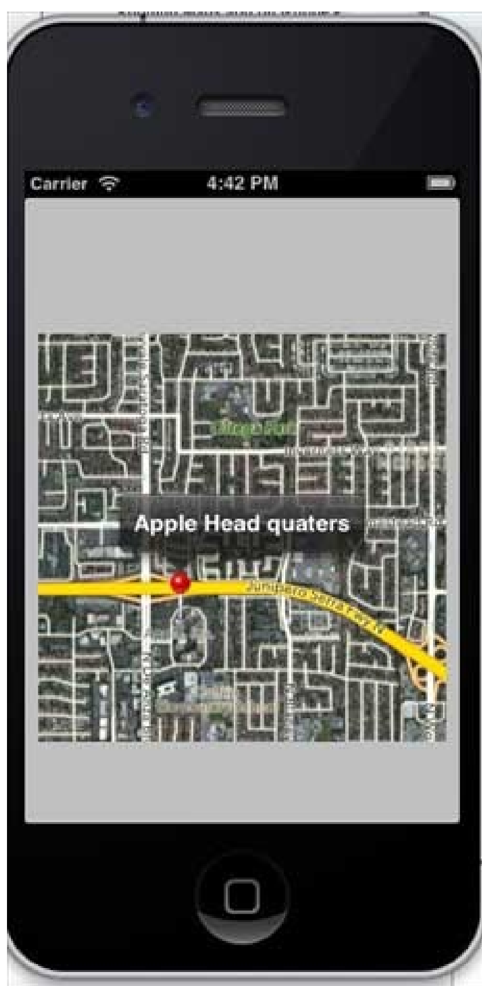
@end
```

输出

运行应用程序时，输出结果如下



当我们向上滚动地图时，输出结果如下


Carrier
4:42 PM
Apple Head quaters

IOS应用内购买

简介

应用程序内购买是应用程序用于购买额外内容或升级功能。

实例步骤

1.在 iTunes 连接中请确保拥有一个唯一的 App ID（unique App ID ），当创建捆绑的ID（ bundle ID）应用程序更新时，代码会以相应的配置文件签名在Xcode上

2.创建新的应用程序和更新应用程序信息。你可以知道更多有关的，在苹果的 添加新的应用程序 文档中

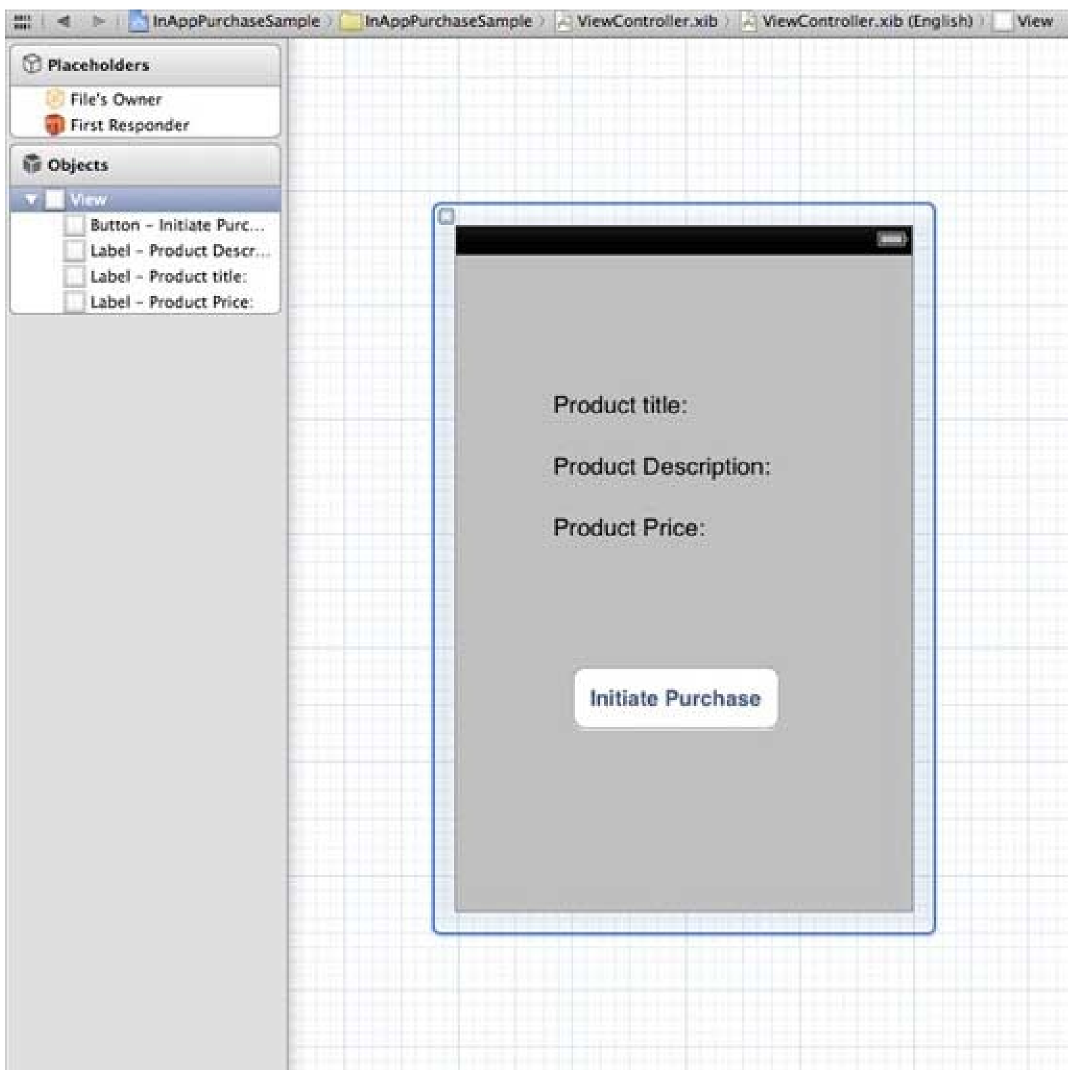
3.在应用程序页的管理应用程序(Manage In-App Purchase)中，为app内付费添加新产品

4.确保设置的应用程序为的银行详细。需要将其设置为在应用程序内购买（In-App purchase）。此外在 iTunes 中使用管理用户（Manage Users）选项，创建一个测试用户帐户连接您的应用程序的页。

5.下一步是与处理代码和为我们在应用程序内购买创建有关的 UI。

6.创建一个单一的视图应用程序，并在 iTunes 中指定的标识符连接输入捆绑标识符

7.更新ViewController.xib ，如下所示



8.为三个标签创建IBOutlets，且将按钮分别命名为 productTitleLabel、productDescriptionLabel、 productPriceLabel 和 purchaseButton

9.选择项目文件，然后选择目标，然后添加StoreKit.framework

10.更新ViewController.h， 如下所示

```
#import <UIKit/UIKit.h>
#import <StoreKit/StoreKit.h>

@interface ViewController : UIViewController<
SKProductsRequestDelegate,SKPaymentTransactionObserver>
{
    SKProductsRequest *productsRequest;
    NSArray *validProducts;
    UIActivityIndicatorView *activityIndicatorView;
    IBOutlet UILabel *productTitleLabel;
    IBOutlet UILabel *productDescriptionLabel;
    IBOutlet UILabel *productPriceLabel;
    IBOutlet UIButton *purchaseButton;
}
- (void)fetchAvailableProducts;
- (BOOL)canMakePurchases;
- (void)purchaseMyProduct:(SKProduct*)product;
- (IBAction)purchase:(id)sender;

@end
```

11.更新ViewController.m， 如下所示

```
#import "ViewController.h"
#define kTutorialPointProductID
@"com.tutorialPoints.testApp.testProduct"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Adding activity indicator
    activityIndicatorView = [[UIActivityIndicatorView alloc]
    initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhit
    activityIndicatorView.center = self.view.center;
    [activityIndicatorView hidesWhenStopped];
    [self.view addSubview:activityIndicatorView];
    [activityIndicatorView startAnimating];
    //Hide purchase button initially
    purchaseButton.hidden = YES;
    [self fetchAvailableProducts];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
```

```
    // Dispose of any resources that can be recreated.
}

-(void)fetchAvailableProducts{
    NSSet *productIdentifiers = [NSSet
    setWithObjects:kTutorialPointProductID,nil];
    productsRequest = [[SKProductsRequest alloc]
    initWithProductIdentifiers:productIdentifiers];
    productsRequest.delegate = self;
    [productsRequest start];
}

- (BOOL)canMakePurchases
{
    return [SKPaymentQueue canMakePayments];
}
- (void)purchaseMyProduct:(SKProduct*)product{
    if ([self canMakePurchases]) {
        SKPayment *payment = [SKPayment paymentWithProduct:product]
        [[SKPaymentQueue defaultQueue] addTransactionObserver:self]
        [[SKPaymentQueue defaultQueue] addPayment:payment];
    }
    else{
        UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:
        @"Purchases are disabled in your device" message:nil delega
        self cancelButtonTitle:@"Ok" otherButtonTitles: nil];
        [alertView show];
    }
}
-(IBAction)purchase:(id)sender{
    [self purchaseMyProduct:[validProducts objectAtIndex:0]];
    purchaseButton.enabled = NO;
}

#pragma mark StoreKit Delegate

-(void)paymentQueue:(SKPaymentQueue *)queue
 updatedTransactions:(NSArray *)transactions {
    for (SKPaymentTransaction *transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchasing:
                    NSLog(@"Purchasing");
                break;
            case SKPaymentTransactionStatePurchased:
               if ([transaction.payment.productIdentifier
                  isEqualToString:kTutorialPointProductID]) {
                   NSLog(@"Purchased ");
                   UIAlertView *alertView = [[UIAlertView alloc]ini
                   @"Purchase is completed succesfully" message:nil
                   self cancelButtonTitle:@"Ok" otherButtonTitles:
                   [alertView show];
                }
                [[SKPaymentQueue defaultQueue] finishTransaction:tr
```

```
                break;
            case SKPaymentTransactionStateRestored:
                NSLog(@"Restored ");
                [[SKPaymentQueue defaultQueue] finishTransaction:tr
                break;
            case SKPaymentTransactionStateFailed:
                NSLog(@"Purchase failed ");
                break;
            default:
                break;
        }
    }
}

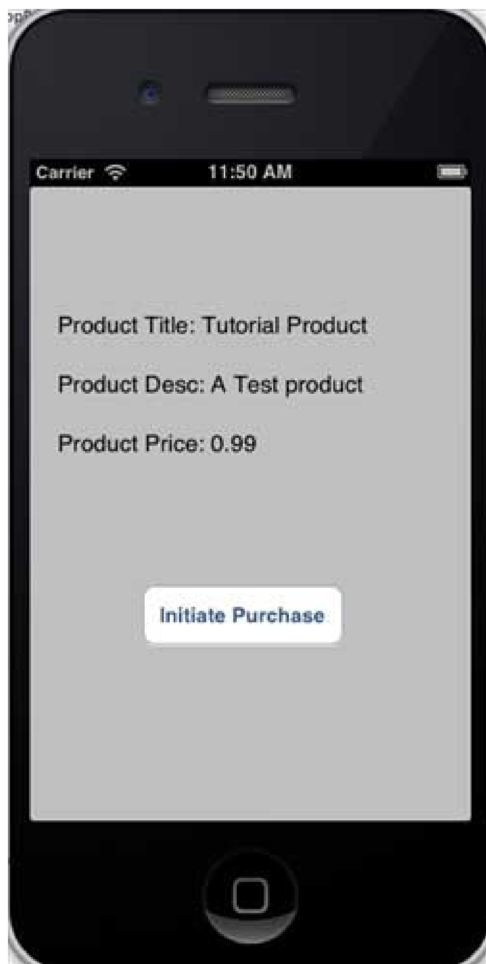
-(void)productsRequest:(SKProductsRequest *)request
 didReceiveResponse:(SKProductsResponse *)response
{
    SKProduct *validProduct = nil;
    int count = [response.products count];
    if (count>0) {
        validProducts = response.products;
        validProduct = [response.products objectAtIndex:0];
        if ([validProduct.productIdentifier
           isEqualToString:kTutorialPointProductID]) {
            [productTitleLabel setText:[NSString stringWithFormat:
            @"Product Title: %@",validProduct.localizedTitle]];
            [productDescriptionLabel setText:[NSString stringWithFo
            @"Product Desc: %@",validProduct.localizedDescription]]
            [productPriceLabel setText:[NSString stringWithFormat:
            @"Product Price: %@",validProduct.price]];
        }
    } else {
        UIAlertView *tmp = [[UIAlertView alloc]
                           initWithTitle:@"Not Available"
                           message:@"No products to purchase"
                           delegate:self
                           cancelButtonTitle:nil
                           otherButtonTitles:@"Ok", nil];
        [tmp show];
    }
    [activityIndicatorView stopAnimating];
    purchaseButton.hidden = NO;
}

@end
```

注意： 需要修改你创建In-App Pur（应用内购买）的 kTutorialPointProductID 。通过修改fetchAvailableProducts产品标识符的 NSSet, 你可以添加多个产品。

输出

运行该应用程序,输出结果如下



确保已经中登录。单击购买选择现有的Apple ID。输入有效的测试帐户的用户名和密码。几秒钟后，显示下面的信息



一旦产品成功购买，将获得以下信息。可以在显示此信息的地方，更新应用功能相关的代码


Carrier
11:55 AM
Product Title: Tutorial Product
Product Desc: A Test product
Purchase is completed succesfully
Ok
Initiate Purchase

IOS iAD整合

简介

IAD是苹果推出的广告平台，它可以帮助开发者从应用程序中获取收入。

实例步骤

1. 创建一个简单的View based application

2. 选择项目文件，然后选择目标，然后选择框架并添加 iAd.framework。

3. 更新 ViewController.h 如下所示

```
#import <UIKit/UIKit.h>
#import <iAd/iAd.h>
@interface ViewController : UIViewController<ADBannerViewDelegate>
{
    ADBannerView *bannerView;
}
@end
```

4. 更新ViewController.m， 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    bannerView = [[ADBannerView alloc]initWithFrame:
    CGRectMake(0, 0, 320, 50)];
    // Optional to set background color to clear color
    [bannerView setBackgroundColor:[UIColor clearColor]];
    [self.view addSubview: bannerView];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - AdViewDelegates

-(void)bannerView:(ADBannerView *)banner
 didFailToReceiveAdWithError:(NSError *)error{
    NSLog(@"Error loading");
}

-(void)bannerViewDidLoadAd:(ADBannerView *)banner{
    NSLog(@"Ad loaded");
}
-(void)bannerViewWillLoadAd:(ADBannerView *)banner{
    NSLog(@"Ad will load");
}
-(void)bannerViewActionDidFinish:(ADBannerView *)banner{
    NSLog(@"Ad did finish");

}
@end
```

输出

运行该应用程序,得到如下输出结果:


Carrier
4:28 PM
$1234
Easily generate
more revenue from
your application.
Find Out How
iAd

IOS GameKit

简介

GameKit是iOS SDK中一个常用的框架。其核心功能有3个：

- 交互游戏平台Game Center,
- P2P设备通讯功能
- In-Game Voice。

实例步骤

1.在链接 iTunes 时请确保拥有一个唯一的 App ID（ unique App ID），App ID在我们应用程序更新 bundle ID时及在Xcode代码签名与相应的配置文件需要使用到。

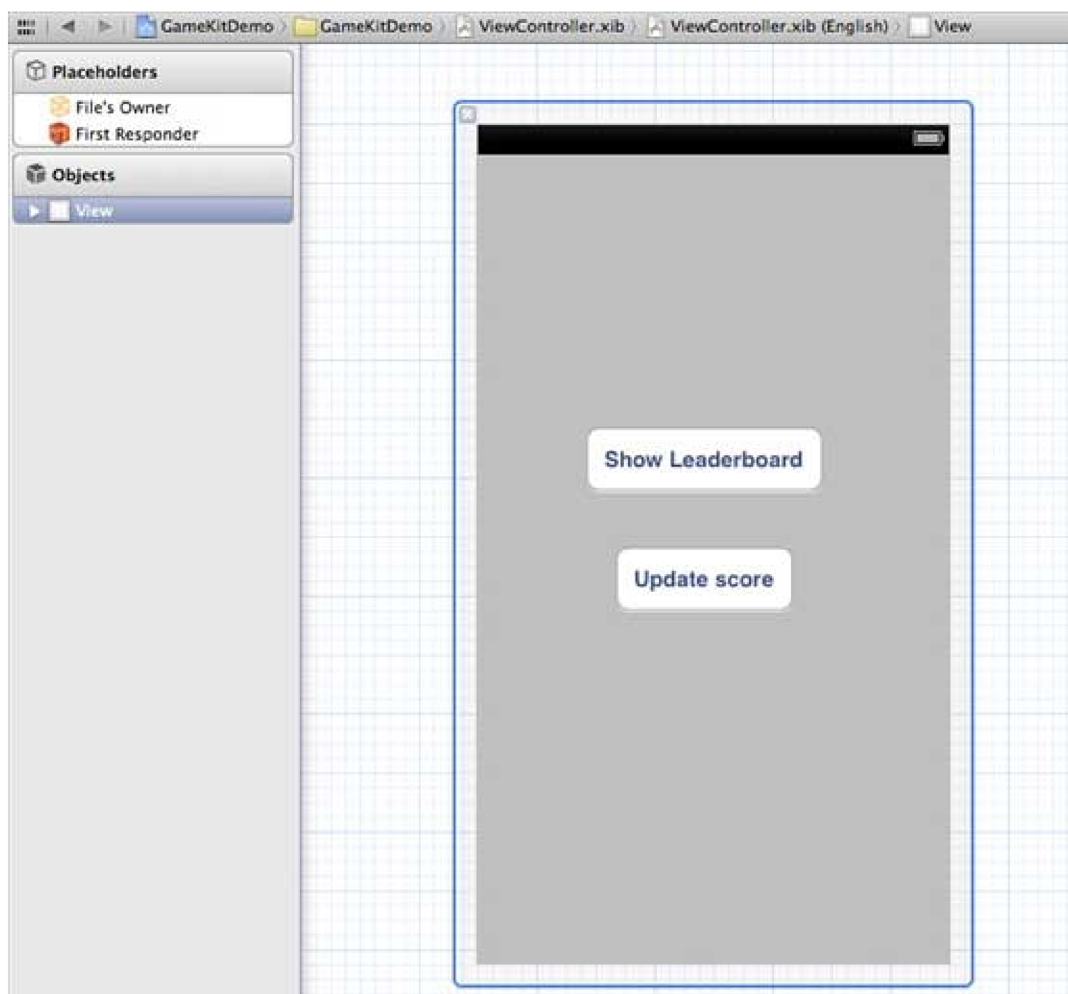
2.创建新的应用程序和更新应用程序信息。在添加新的应用程序文档可以了解更多有关信息。

3.打开你申请的application,点击Manage Game Center选项。进入后点击Enable Game Center使你的Game Center生效。接下来设置自己的Leaderboard和Achievements。

4.下一步涉及处理代码，并为我们的应用程序创建用户界面。

5.创建一个single view application，并输入 bundle identifier 。

6.更新 ViewController.xib，如下所示



7.选择项目文件，然后选择目标，然后添加GameKit.framework

8.为已添加的按钮创建IBActions

9.更新ViewController.h文件，如下所示

```
#import <UIKit/UIKit.h>
#import <GameKit/GameKit.h>

@interface ViewController : UIViewController
<GKLeaderboardViewControllerDelegate>

-(IBAction)updateScore:(id)sender;
-(IBAction)showLeaderBoard:(id)sender;

@end
```

10.更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()
```

```
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    if([GKLocalPlayer localPlayer].authenticated == NO)
    {
      [[GKLocalPlayer localPlayer]
      authenticateWithCompletionHandler:^(NSError *error)
      {
         NSLog(@"Error%@",error);
      }];
    }
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void) updateScore: (int64_t) score
forLeaderboardID: (NSString*) category
{
    GKScore *scoreObj = [[GKScore alloc]
    initWithCategory:category];
    scoreObj.value = score;
    scoreObj.context = 0;
    [scoreObj reportScoreWithCompletionHandler:^(NSError *error) {
        // Completion code can be added here
        UIAlertView *alert = [[UIAlertView alloc]
        initWithTitle:nil message:@"Score Updated Succesfully"
        delegate:self cancelButtonTitle:@"Ok" otherButtonTitles: ni
        [alert show];

    }];
}
-(IBAction)updateScore:(id)sender{
    [self updateScore:200 forLeaderboardID:@"tutorialsPoint"];
}
-(IBAction)showLeaderBoard:(id)sender{
    GKLeaderboardViewController *leaderboardViewController =
    [[GKLeaderboardViewController alloc] init];
    leaderboardViewController.leaderboardDelegate = self;
    [self presentModalViewController:
    leaderboardViewController animated:YES];

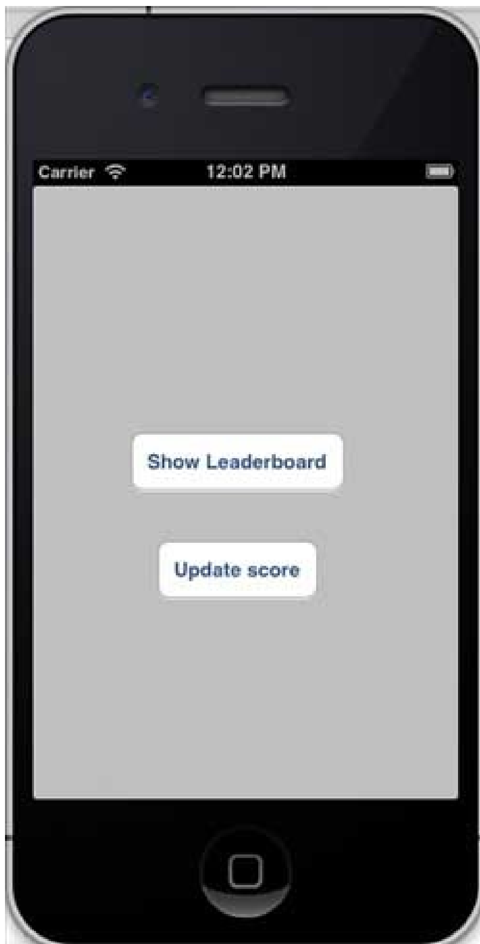
}
#pragma mark - Gamekit delegates
- (void)leaderboardViewControllerDidFinish:
(GKLeaderboardViewController *)viewController{
    [self dismissModalViewControllerAnimated:YES];
```

```
}

@end
```

输出

运行该应用程序，输出结果如下



当我们单击显示排行榜时，屏幕显示如下:



当我们点击更新分数，比分将被更新到我们排行榜上，我们会得到一个信息，如下图所示


Carrier
12:05 PM
Score Updated Succesfully
Ok
Update score

IOS 故事板(Storyboards)

简介

Storyboards在 iOS 5 中才有介绍，当我们用Storyboards时，部署目标应该是iOS5.0或更高版本。

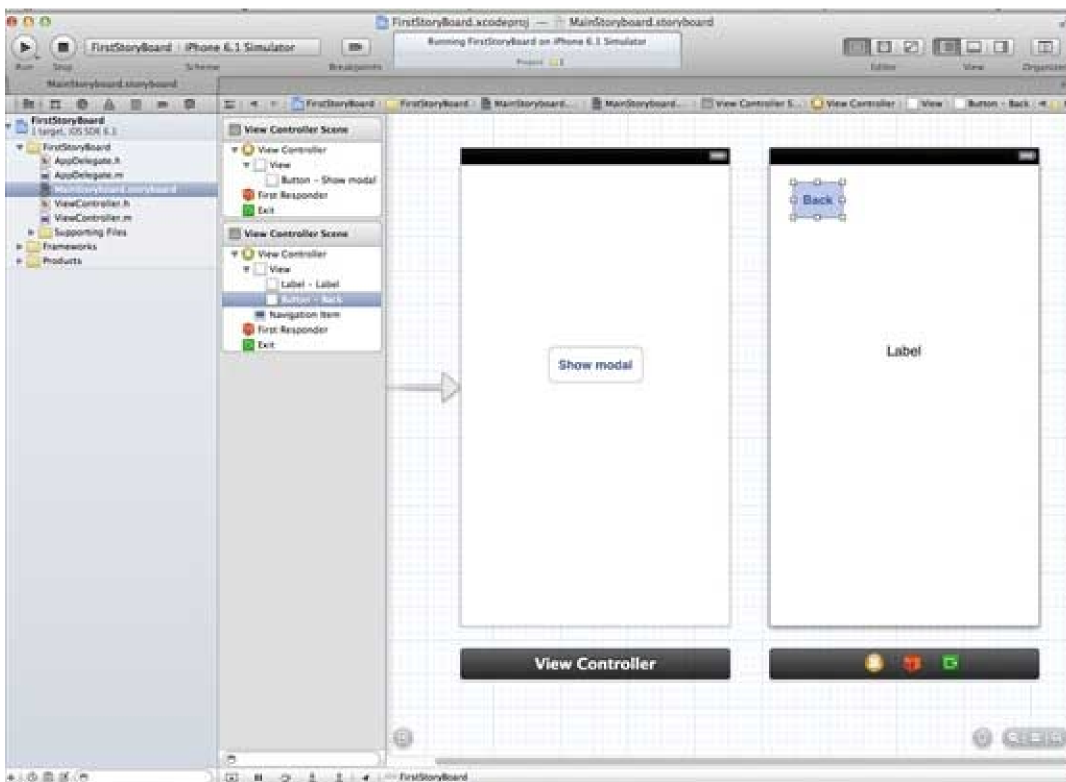
Storyboards 帮助我们了解视觉流动的画面，在界面为

MainStoryboard.storyboard下创建所有应用程序屏幕。

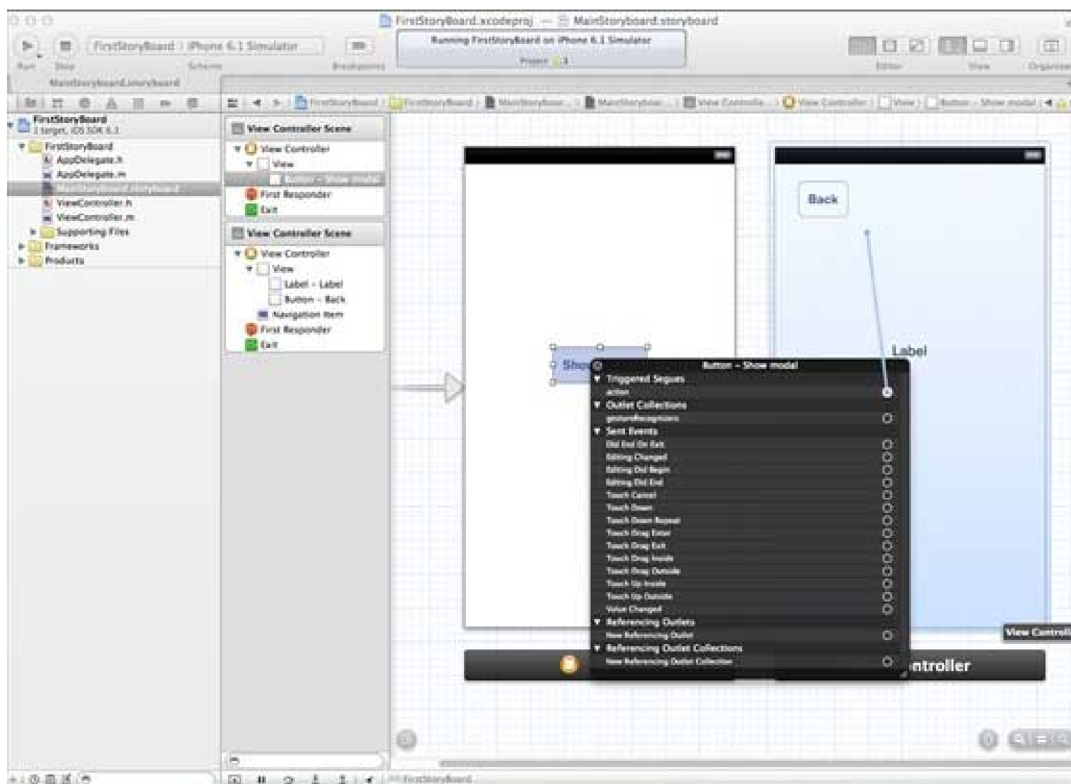
实例步骤

1. 创建一个single view application，创建应用程序时选择 storyboard 复选框。

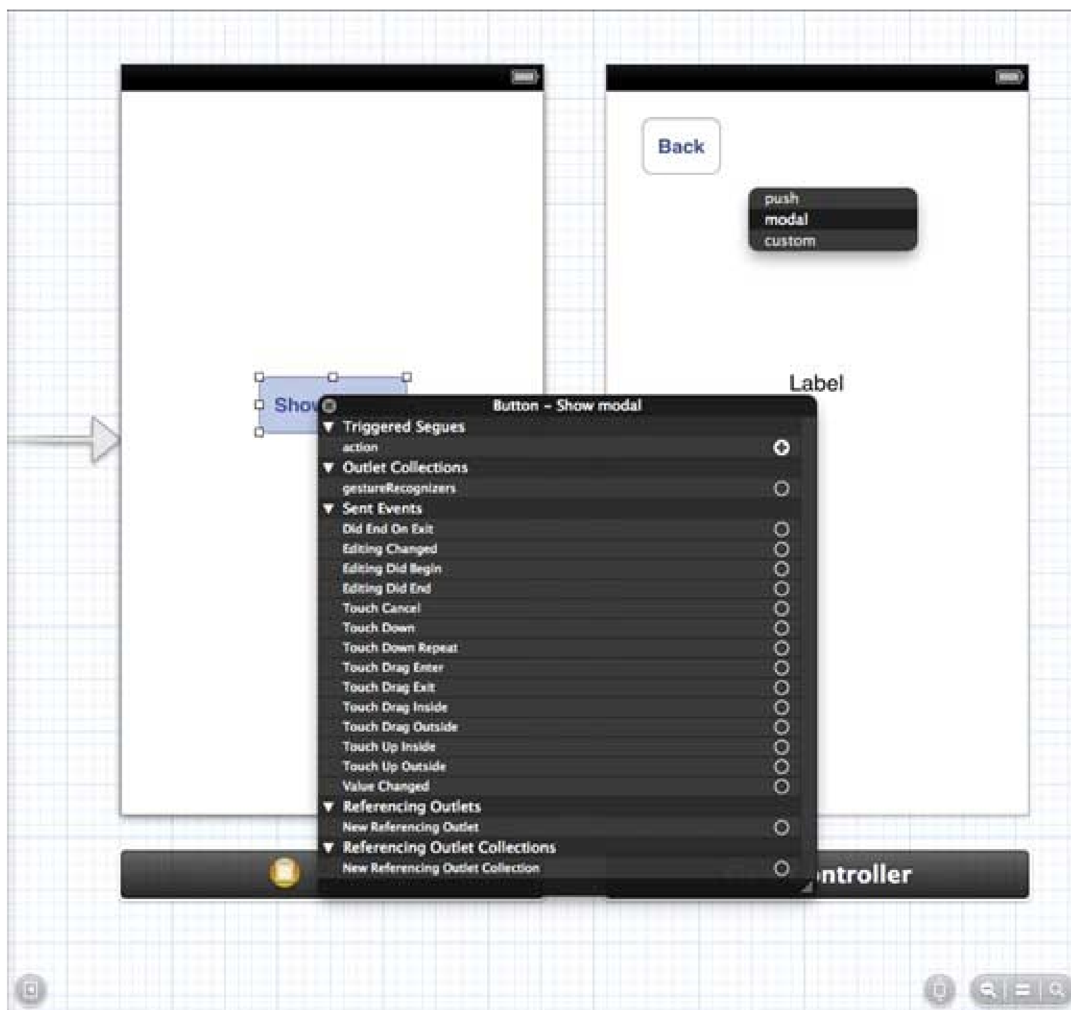
2. 选择MainStoryboard.storyboard，在这里你可以找到单一视图控制器。添加一个视图控制器，更新视图控制器，如下所示



3.连接两个视图控制器。右键单击"show modal（显示模式）"按钮，在左侧视图控制器将其拖动到右视视图控制器中,如下图所示：



4.现在从如下所示的三个显示选项中选择modal(模态)



5.更新 ViewController.h 如下所示

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

-(IBAction)done:(UIStoryboardSegue *)seque;

@end
```

6.更新 ViewController.m 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

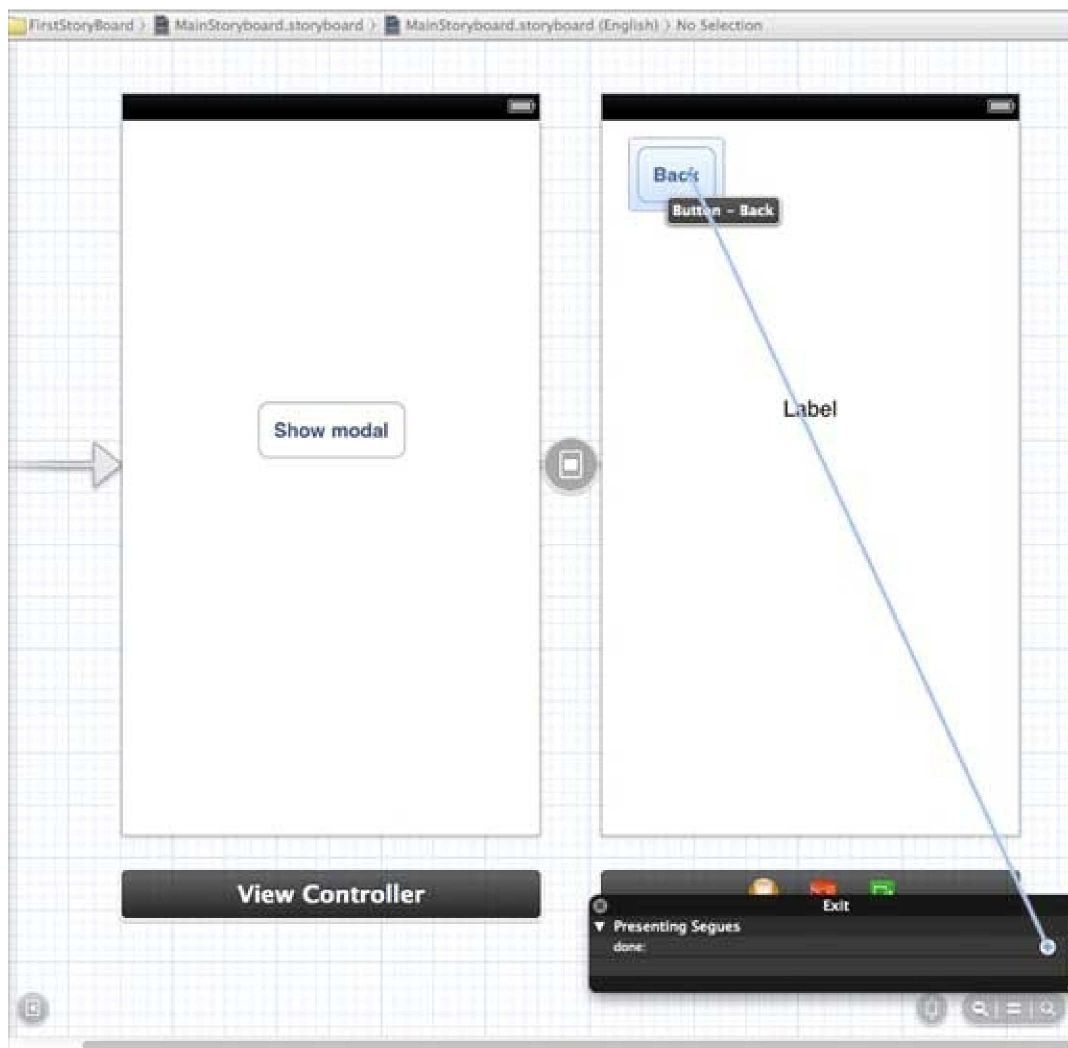
@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
-(IBAction)done:(UIStoryboardSegue *)seque{
    [self.navigationController popViewControllerAnimated:YES];
}

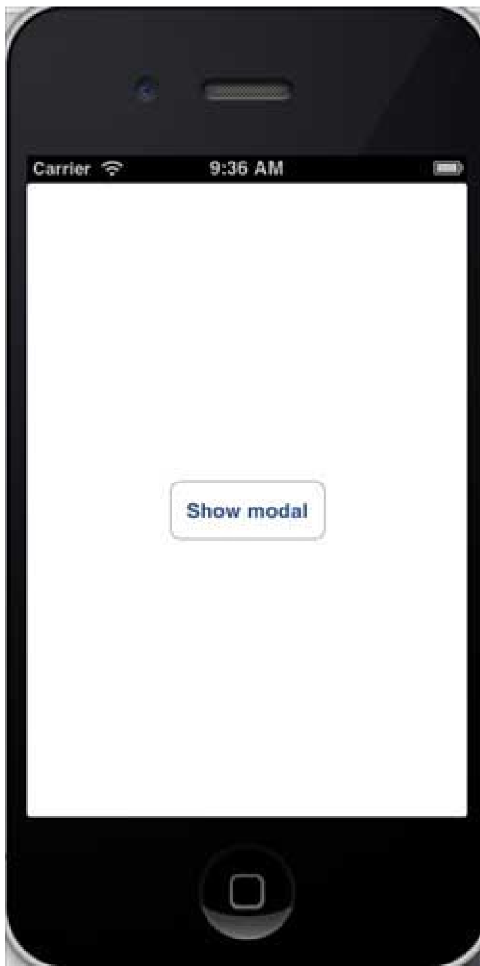
@end
```

7.选择"MainStoryboard.storyboard"，并右键点击"Exit "按钮，在右侧视图控制器中选择和连接后退按钮，如下图所示



输出

在iPhone设备中运行该应用程序,得到如下输出结果



现在，选择显示模式，将得到下面的输出结果


Carrier
9:36 AM
Back
Label

IOS自动布局

简介

自动布局在iOS 6.0中引入，仅可以支持IOS6.0 及 更高版本。它可以帮助我们创建用于多个种设备的界面。

实例步骤

1.创建一个简单的 View based application

2.修改 ViewController.m 的文件内容，如下所示

```
#import "ViewController.h"
@interface ViewController ()
@property (nonatomic, strong) UIButton *leftButton;
@property (nonatomic, strong) UIButton *rightButton;
@property (nonatomic, strong) UITextField *textfield;

@end
@implementation ViewController

- (void)viewDidLoad{
    [super viewDidLoad];
    UIView *superview = self.view;
    /*1\. Create leftButton and add to our view*/
    self.leftButton = [UIButton buttonWithType:UIButtonTypeRoundedR
    self.leftButton.translatesAutoresizingMaskIntoConstraints = NO;
    [self.leftButton setTitle:@"LeftButton" forState:UIControlState
    [self.view addSubview:self.leftButton];
    /* 2\. Constraint to position LeftButton's X*/
    NSLayoutConstraint *leftButtonXConstraint = [NSLayoutConstraint
    constraintWithItem:self.leftButton attribute:NSLayoutAttributeC
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterX multiplier:1.0 constant:-60.0f];
    /* 3\. Constraint to position LeftButton's Y*/
    NSLayoutConstraint *leftButtonYConstraint = [NSLayoutConstraint
    constraintWithItem:self.leftButton attribute:NSLayoutAttributeC
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeCenterY multiplier:1.0f constant:0.0f];
    /* 4\. Add the constraints to button's superview*/
    [superview addConstraints:@[ leftButtonXConstraint,
    leftButtonYConstraint]];
    /*5\. Create rightButton and add to our view*/
    self.rightButton = [UIButton buttonWithType:UIButtonTypeRounded
    self.rightButton.translatesAutoresizingMaskIntoConstraints = NO
    [self.rightButton setTitle:@"RightButton" forState:UIControlSta
```

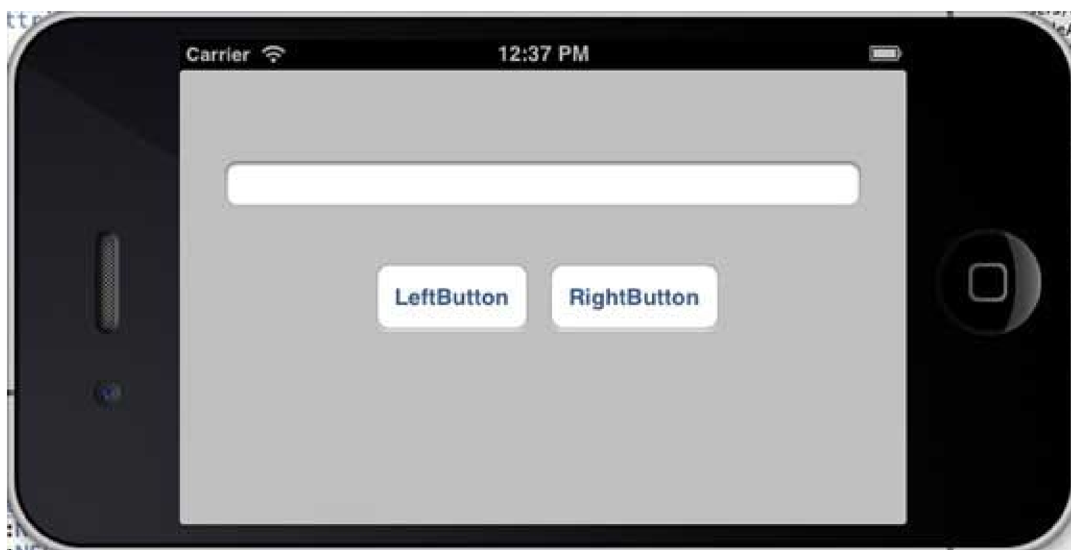
```
    [self.view addSubview:self.rightButton];
    /*6\. Constraint to position RightButton's X*/
    NSLayoutConstraint *rightButtonXConstraint = [NSLayoutConstrain
    constraintWithItem:self.rightButton attribute:NSLayoutAttribute
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterX multiplier:1.0 constant:60.0f];
    /*7\. Constraint to position RightButton's Y*/
    rightButtonXConstraint.priority = UILayoutPriorityDefaultHigh;
    NSLayoutConstraint *centerYMyConstraint = [NSLayoutConstraint
    constraintWithItem:self.rightButton attribute:NSLayoutAttribute
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterY multiplier:1.0f constant:0.0f];
    [superview addConstraints:@[centerYMyConstraint,
    rightButtonXConstraint]];
   //8\. Add Text field
    self.textfield = [[UITextField alloc]initWithFrame:
    CGRectMake(0, 100, 100, 30)];
    self.textfield.borderStyle = UITextBorderStyleRoundedRect;
    self.textfield.translatesAutoresizingMaskIntoConstraints = NO;
    [self.view addSubview:self.textfield];
    //9\. Text field Constraints
    NSLayoutConstraint *textFieldTopConstraint = [NSLayoutConstrain
    constraintWithItem:self.textfield attribute:NSLayoutAttributeTo
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview
    attribute:NSLayoutAttributeTop multiplier:1.0 constant:60.0f];
    NSLayoutConstraint *textFieldBottomConstraint = [NSLayoutConstr
    constraintWithItem:self.textfield attribute:NSLayoutAttributeTo
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:self.rightB
    attribute:NSLayoutAttributeTop multiplier:0.8 constant:-60.0f];
    NSLayoutConstraint *textFieldLeftConstraint = [NSLayoutConstrai
    constraintWithItem:self.textfield attribute:NSLayoutAttributeLe
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeLeft multiplier:1.0 constant:30.0f];
    NSLayoutConstraint *textFieldRightConstraint = [NSLayoutConstra
    constraintWithItem:self.textfield attribute:NSLayoutAttributeRi
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeRight multiplier:1.0 constant:-30.0f];
    [superview addConstraints:@[textFieldBottomConstraint ,
    textFieldLeftConstraint, textFieldRightConstraint,
    textFieldTopConstraint]];
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end
```

输出

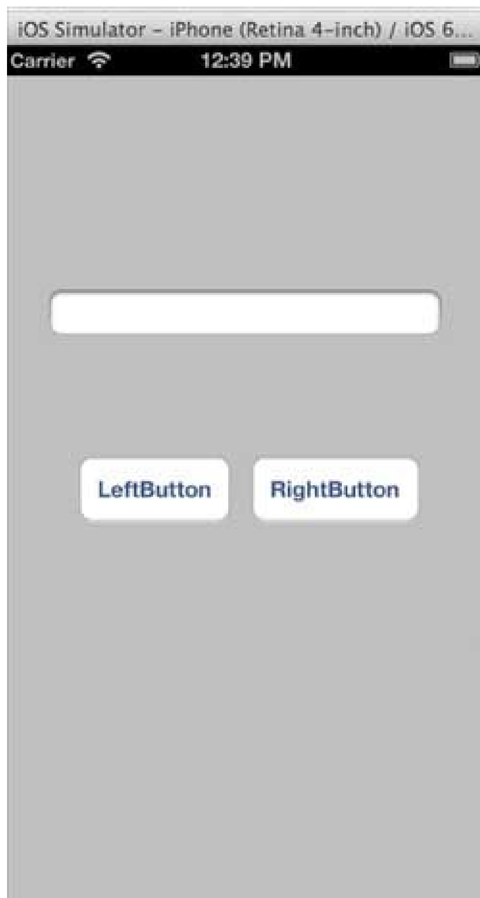
运行应用程序，在 iPhone 模拟器上会有下面的输出结果



当我们更改模拟器为横向的方向时，输出结果如下



我们在 iPhone 5 模拟器上运行同一应用程序时,输出结果如下



当我们更改模拟器为横向的方向时，输出结果如下:



IOS-Twitter和Facebook

简介

Twitter已经整合到iOS5.0，而Facebook已经被集成在 iOS 6.0中。本教程的重点讲解如何利用苹果提供的类在iOS5.0和iOS6.0中部署Twitter和Facebook。

实例步骤

1. 创建一个简单View based application

2. 选择项目文件，然后选择"targets(目标)"，然后在 choose frameworks（选择框架）中添加Social.framework 和 Accounts.framework

3. 添加两个名为facebookPost 和 twitterPost的按钮，并为他们创建 ibActions。

4. 更新 ViewController.h 如下

```
#import <Social/Social.h>
#import <Accounts/Accounts.h>
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

-(IBAction)twitterPost:(id)sender;
-(IBAction)facebookPost:(id)sender;

@end
```

5. 更新ViewController.m， 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
```

```
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(IBAction)facebookPost:(id)sender{

   SLComposeViewController *controller = [SLComposeViewController
   composeViewControllerForServiceType:SLServiceTypeFacebook];
   SLComposeViewControllerCompletionHandler myBlock =
   ^(SLComposeViewControllerResult result){
          if (result == SLComposeViewControllerResultCancelled)
          {
            NSLog(@"Cancelled");
          }
          else
          {
            NSLog(@"Done");
          }
          [controller dismissViewControllerAnimated:YES completion
        };
   controller.completionHandler =myBlock;
   //Adding the Text to the facebook post value from iOS
   [controller setInitialText:@"My test post"];
   //Adding the URL to the facebook post value from iOS
   [controller addURL:[NSURL URLWithString:@"http://www.test.com"]]
   //Adding the Text to the facebook post value from iOS
   [self presentViewController:controller animated:YES completion:n
}

-(IBAction)twitterPost:(id)sender{
    SLComposeViewController *tweetSheet = [SLComposeViewController
    composeViewControllerForServiceType:SLServiceTypeTwitter];
    [tweetSheet setInitialText:@"My test tweet"];
    [self presentModalViewController:tweetSheet animated:YES];
}

@end
```

输出

运行该应用程序并单击 facebookPost 时我们将获得以下输出



当我们单击 twitterPost 时，我们将获得以下输出


Carrier
4:59 PM
Cancel
Tweet
Send
My test tweet
Add Location
127
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123
space
@
#

IOS内存管理

简介

iOS下内存管理的基本思想就是引用计数，通过对象的引用计数来对内存对象的生命周期进行控制。具体到编程时间方面，主要有两种方式：

1：MRR（manual retain-release），人工引用计数，对象的生成、销毁、引用计数的变化都是由开发人员来完成。

2：ARC（Automatic Reference Counting），自动引用计数，只负责对象的生成，其他过程开发人员不再需要关心其销毁，使用方式类似于垃圾回收，但其实质还是引用计数。

面临的问题

根据苹果说明文档，面临的两个主要问题是：

释放或覆盖的数据仍然在使用。这将造成内存损坏，通常在应用程序崩溃，或者更糟，损坏用户数据。

不释放不再使用的数据会导致内存泄漏。分配的内存，内存泄漏不会释放，即使它从来没有再次使用。泄漏会导致应用程序的内存使用量日益增加，这反过来又可能会导致系统性能较差或死机。

内存管理规则

我们创建自己的对象，当他们不再需要的时候，释放他们。

保留需要使用的对象。如果没有必要必须释放这些对象。

不要释放我们没有拥有的对象。

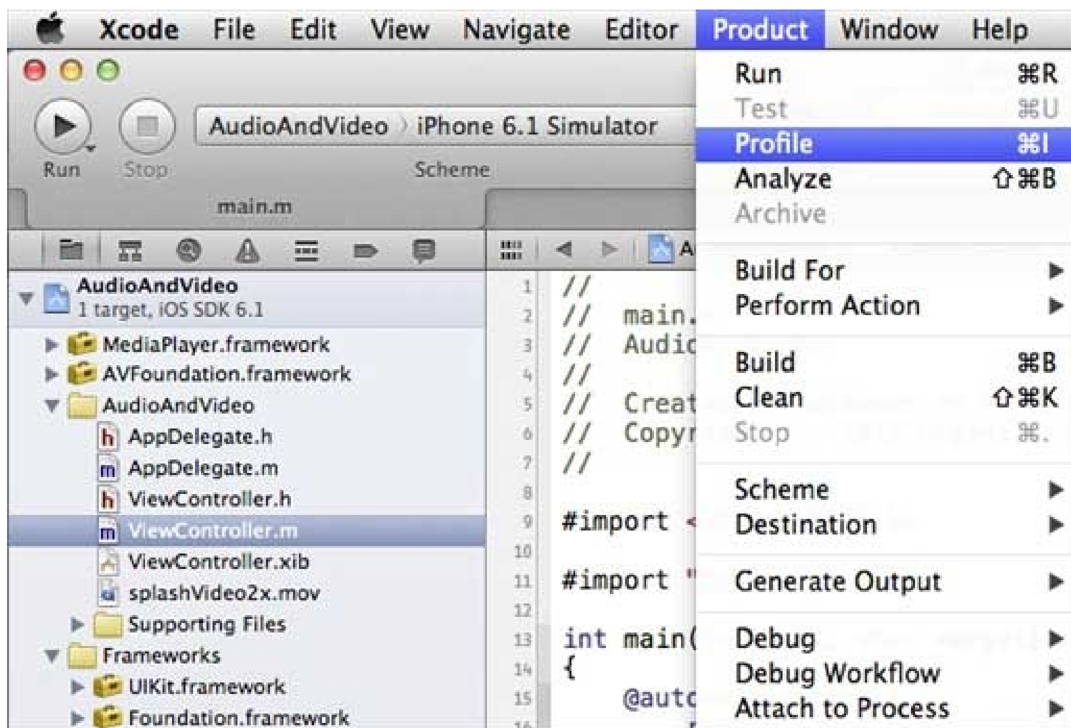
使用内存管理工具

可以用Xcode工具仪器的帮助下分析内存的使用情况。它包括的工具有活动监视器，分配，泄漏，僵尸等

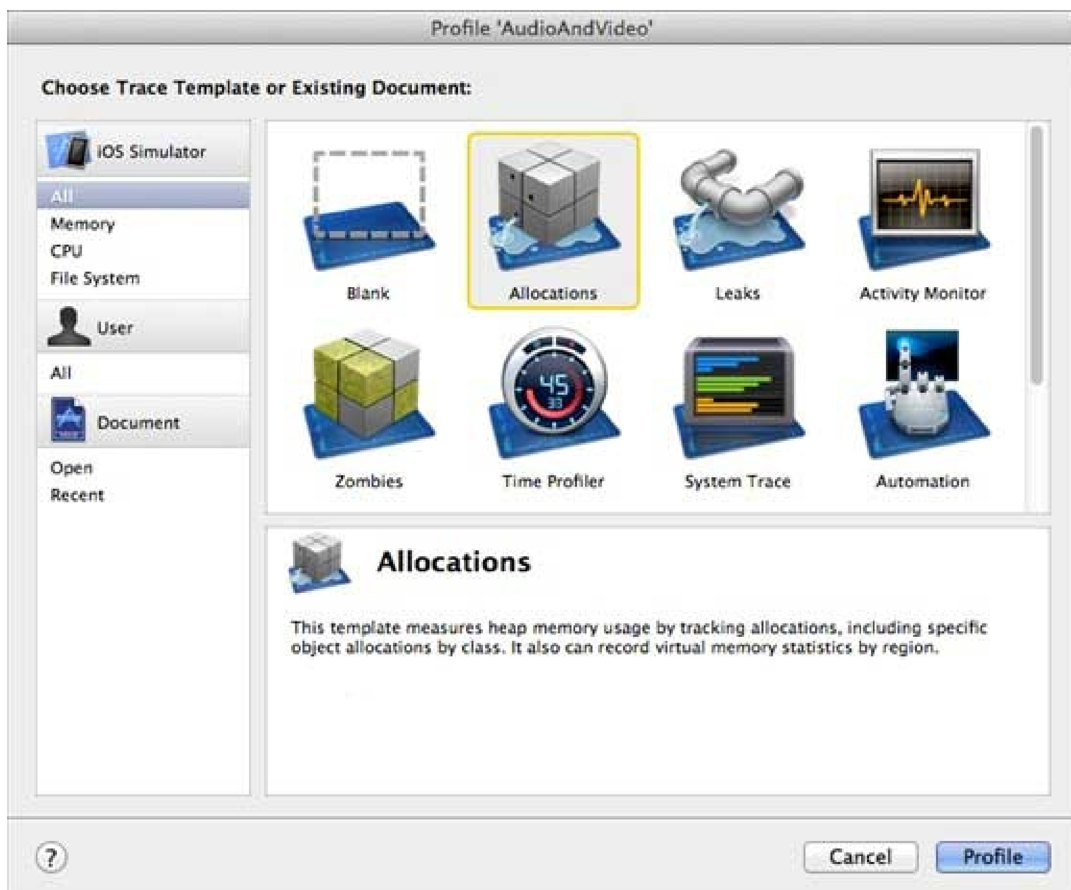
分析内存分配的步骤

1. 打开一个现有的应用程序。

2. 选择产品，配置文件如下所示

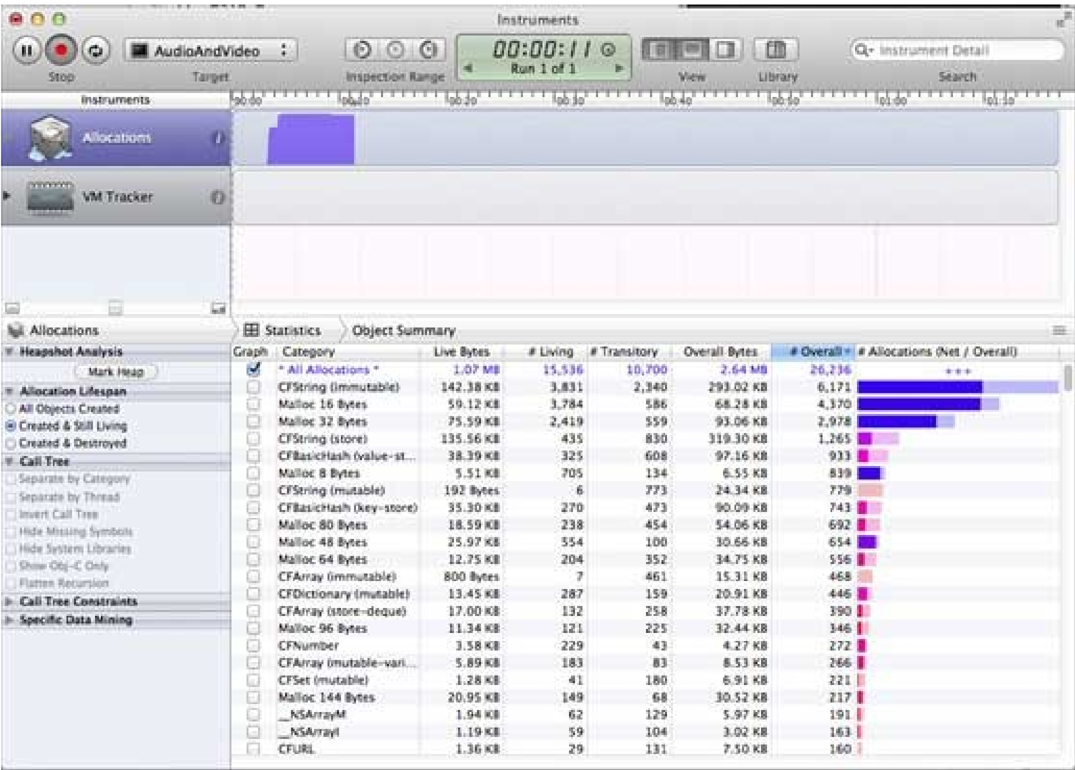


3.在以下界面中选择 Allocations 和 Profile。

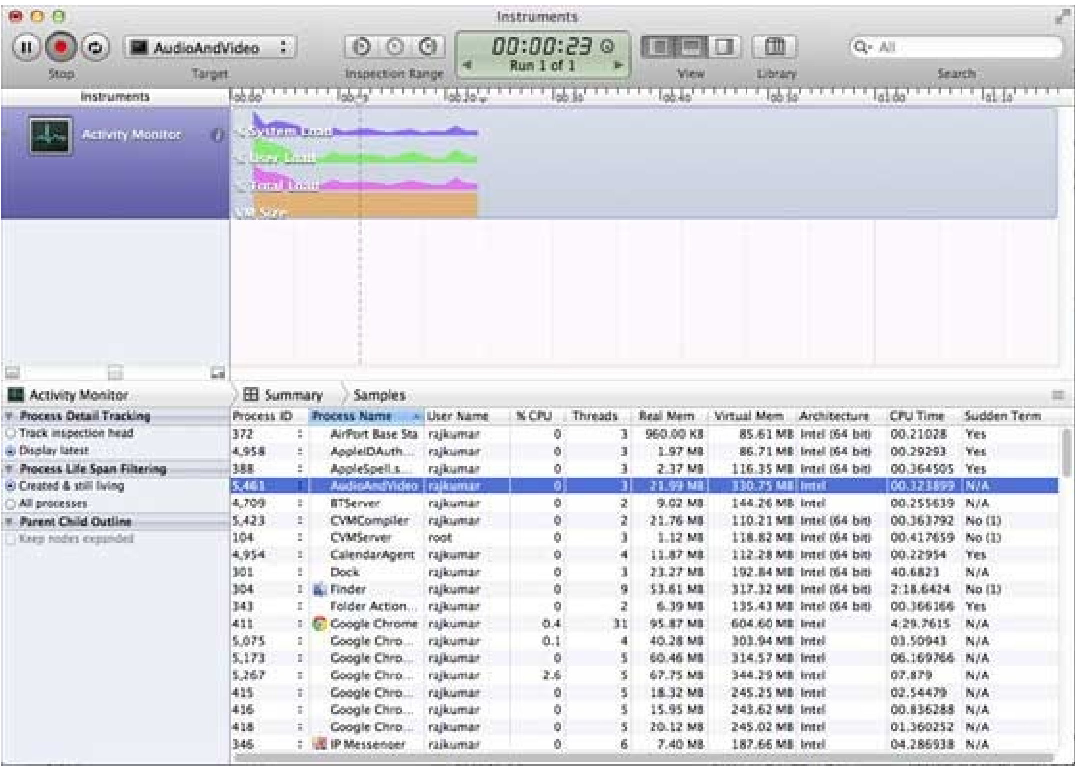


4. 我们可以看到不同对象的内存使用情况

5. 你可以切换视图控制器查看内存是否释放。



6.同样我们可以使用 Activity Monitor 来查看内存在应用程序中的分配的情况。



7. 这些工具可以帮助我们了解内存的使用情况及在什么地方可能发生泄漏。

IOS应用程序调试

简介

当我们做应用程序的时候，可能会犯各种错误，这可能会导致各种不同的错误。因此，为了修复这些错误或缺陷，我们需要来调试应用程序。

选择一个调试器

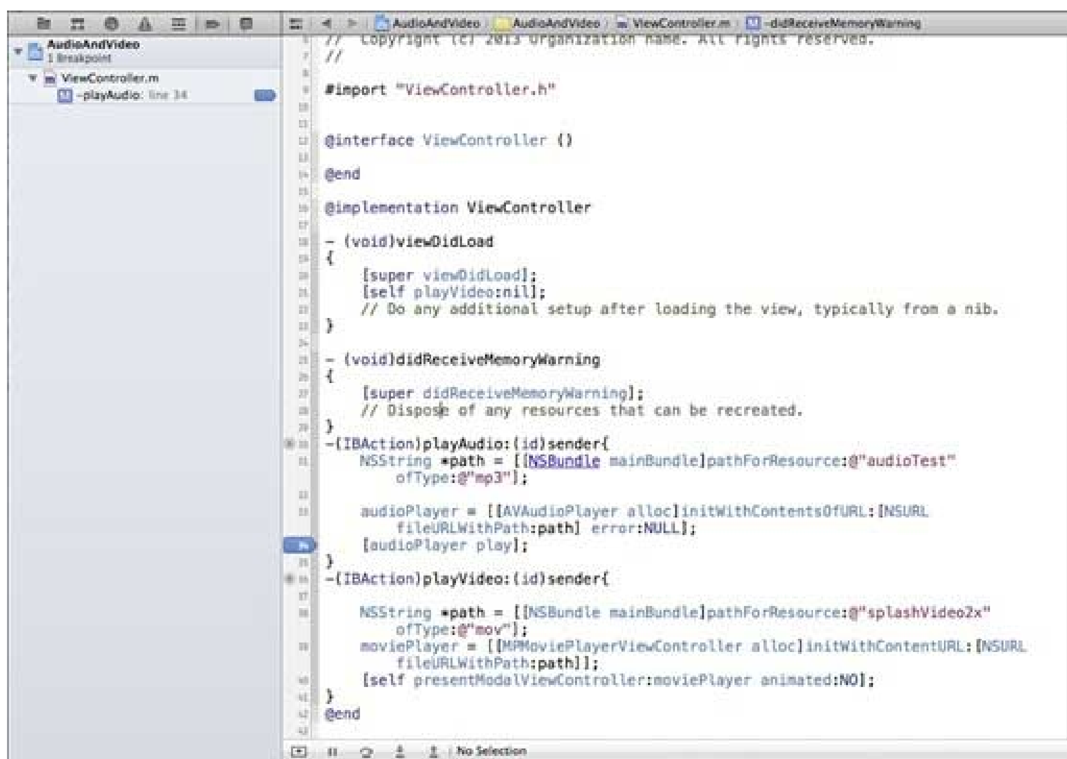
Xcode中调试器即 GDB 和 LLDB 调试器，GDB 是默认的。 LLDB是一个调试器是LLVM开源的编译器项目的一部分。您可以更改调试，编辑活动计划选项。

如何查找编码错误？

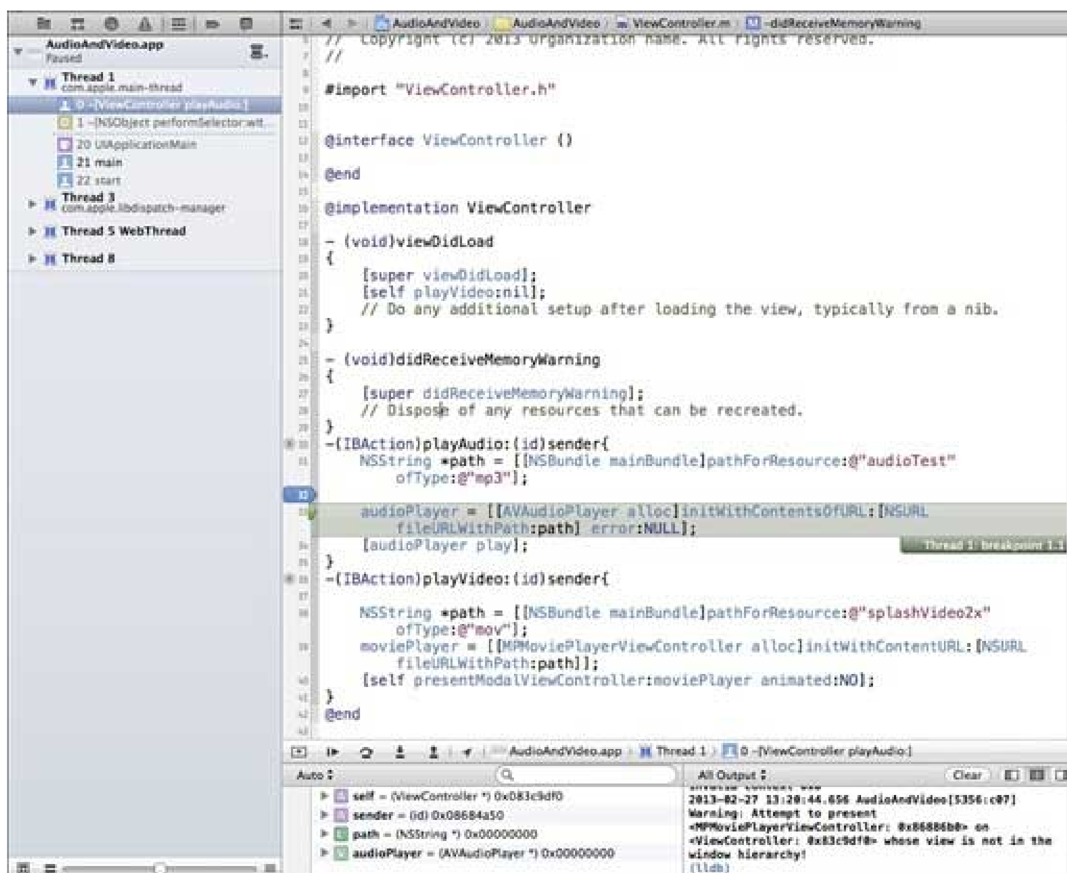
我们只需要建立我们的应用程序，代码被编译器编译，所有的消息，错误和警告将显示以及错误的原因，我们可以纠正他们。可以点击 product，然后点击"分析"，将在应用程序中可能发生的问题。

设置断点

断点帮助我们了解我们的应用程序对象，帮助我们找出许多缺陷，包括逻辑问题的不同状态。我们只需要点击创建一个断点的行号。我们可以通过点击并拖动它删除断点。如下所示



当我们运行应用程序并选择playVideo，按钮的应用程序将被暂停，我们来分析一下我们的应用程序的状态。当断点被触发时，我们将得到一个输出，如下图所示



可以轻松地确定哪个线程触发断点。在底部可以看到对象，如self，sender等，这些持有相应的对象的值，我们可以展开一些这些对象，看看他们每个的状态是什么。

要继续应用程序，我们在调试区选择继续按钮（最左边的按钮），如下图所示。其他选项包括步骤和单步跳过

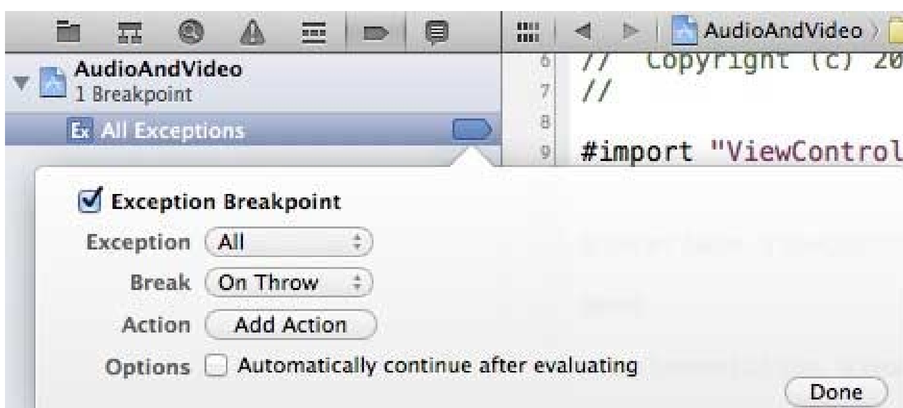


异常断点

我们也有异常断点，触发应用程序停止发生异常的位置。通过选择调试导航后选择"+"按钮，我们可以创建异常断点。将得到下面的窗口



然后，我们需要选择" Exception Breakpoint (添加异常)"断点，它会显示下面的窗口



下一步是什么？

你可以在 Xcode 4 用户指南 知道更多关于调试和其他Xcode功能的知识。

免责声明

W3School提供的内容仅用于培训。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关。W3School简体中文版的所有内容仅供测试，对任何法律问题及风险不承担任何责任。